CREW EXPLORATION VEHICLE (CEV) SKIP ENTRY TRAJECTORY

THESIS

Emre Kaya, First Lieutenant, TuAF

AFIT/GSS/ENY/08-M06

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the U.S. Government.

AFIT/GSS/ENY/08-M06

CREW EXPLORATION VEHICLE (CEV) SKIP ENTRY TRAJECTORY

THESIS

Presented to the Faculty

Department of Aeronautics and Astronautics

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science in Space Systems

Emre Kaya, BS

First Lieutenant, TuAF

March 2008

AFIT/GSS/ENY/08-M06

CREW EXPLORATION VEHICLE (CEV) SKIP ENTRY TRAJECTORY

Emre Kaya, BS

First Lieutenant, TuAF

Approved:

_____// SIGNED //_____	03/13/2008
Kerry Hicks, Lt Col, USAF (Chairman)	Date
_____// SIGNED //_____	03/13/2008
Dr. William E. Wiesel (Member)	Date
_____// SIGNED //_____	03/13/2008
Dr. Jonathan Black (Member)	Date

Acknowledgments

I would like to express my sincere appreciation to my faculty advisor, Lt Col Kerry Hicks, for his guidance and support throughout the course of this thesis effort. His insight and experience on spaceflight and reentry dynamics were invaluable. I would, also, like to thank Turkish Air Force for their great support in every aspect.

Emre Kaya

Table of Contents

Page

List of Figures

Page

List of Tables

Page

Abstract

This research effort develops a program using MATLAB® to solve the equations of motion for the atmospheric reentry of the Crew Exploration Vehicle (CEV) which is assumed to be in the phase of a lunar return trajectory that could be initiated any time during the mission. The essential reason for this research is to find a solution for the problem of an unplanned lunar return in addition to the normal procedures. Unlike Apollo type missions, the CEV would still be able to land on any preplanned available landing sites without any additional delay. In Apollo type missions, the return phase had to be initiated in a restricted time window so that the crew module could enter the atmosphere at the preplanned time and be able to land at the planned landing site. Using skip entry procedures, landing location and time will be more accurate in addition to having the time flexibility for reentry. This MATLAB® program is designed to find the reentry parameters for given landing location according to the current alignment of the moon using a lunar return speed including the atmospheric trajectory of the CEV.

CREW EXPLORATION VEHICLE (CEV) SKIP ENTRY TRAJECTORY

I. Introduction

Background

The renewed interest in human exploration beyond low orbit has led to many different viewpoints on which exploration architecture is appropriate for human missions to the Moon and Mars. In 2004, the President of the United States fundamentally shifted the priorities of America's civil space program with the Vision for Space Exploration (VSE), calling for long-term human exploration of the Moon, Mars and beyond. [1] This program focuses on returning astronauts to the Moon by 2020 with the eventual establishment of a permanent manned station there. Experience gained from human exploration of the Moon is then to be used to prepare for a human mission to Mars. To complete these tasks, a new human exploration vehicle, the Crew Exploration Vehicle (CEV) will be developed. [1]

While numerous exploration architectures exist for a lunar mission, the goal is to come up with a combined moon and possible mars exploration vehicle with a reliable and accurate reentry system. Among these options, most reentry systems require high-speed, aero-assisted deceleration of a crewed vehicle at Earth entry. Having many possible options for the entry system, the selection will have a significant effect on the overall exploration architecture. The entry system is typically carried through an entire mission, and, its mass, size and complexity can have large impact on other architectural elements.

The NASA Exploration Systems Architecture Study (ESAS) selected a CEV similar to the Apollo Program's Command and Service Module, with a crewed command module and an unmanned service module. As seen in Figure 1, the CEV command module will be a scaled version of the Apollo Command Module (CM), maintaining the same outer mold line with a larger radius for more cargo and crew capacity. In addition, the CEV will be required to return safely to land locations during normal operations, as opposed to the ocean landings performed in the Apollo program.

Figure 1. Schematic of CEV CM [10]

Unlike Apollo missions, CEV missions are also required to be flexible for unscheduled mission changes beside the normal operations. In such an emergency, or after an early mission completion, the CEV and its crew will be capable of starting the

lunar return procedures and still be able to land at one of the preplanned available landing sites.

Although this mission is carried out by the entry guidance system integrated in the flight computer of the CEV, early selection of the reentry coordinates and parameters is still needed to save propellant that can be used for the attitude control and adjustment. The third body effect is minor in this near Earth operations; however, its perturbations will still need to be countered during the trajectory in order to be able to meet the right entry parameters. Therefore, minimal propellant consumption also is important for successful mission accomplishment.

The Apollo program entry guidance contained a long-range option to provide an abort mode in the event of poor weather conditions at the primary landing site. Moderate L/D blunt body entry vehicles, such as the CEV, can easily achieve long-range entries by employing a skipping entry trajectory. When performing a skipping entry, the vehicle enters the atmosphere and begins to decelerate. The vehicle then uses aerodynamic forces to execute a pull-up maneuver, lofting the vehicle to higher altitudes, possibly exiting the atmosphere.[2] However, enough energy is dissipated during the first atmospheric flight segment to ensure that the vehicle will enter the atmosphere a second time, at a point significantly farther downrange than the initial entry point. After the second entry, the vehicle proceeds to the surface. A longer-range trajectory is achieved in this manner, as shown in Figure 2.

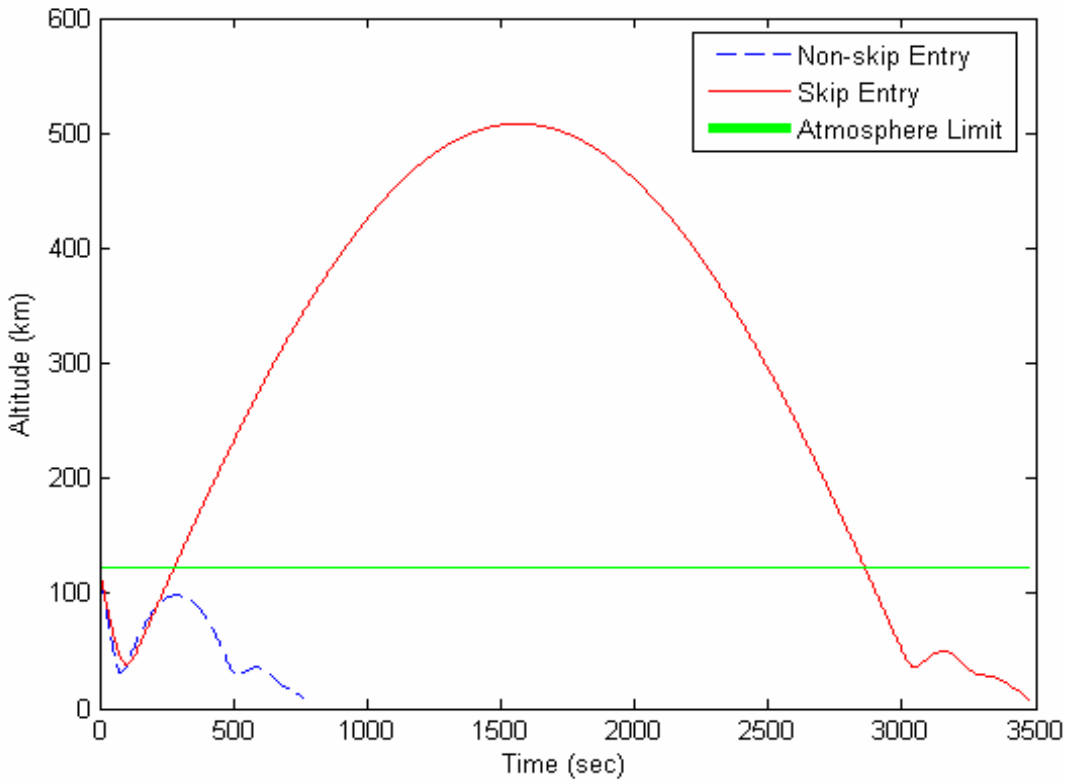

Figure 2. Skip and Non-skip Entry Trajectories (Altitude vs. Time)

In addition, the Apollo CM guidance was designed to allow a maximum deceleration of 12g during nominal entry. Typical Apollo missions reached peak decelerations over 6.5g during entry with the help of "double dip reentry." [13] Compared to non-skip entry conditions, lower g load values are reached in the skip entry trajectory due to the large energy dissipation during the first atmospheric flight segment. In a double dip entry, the vehicle does not complete a skip entry but loses its excess energy by accomplishing the first part of the skip but never leaving the atmosphere as seen in Figure 3. To do this, the vehicle rolls over after the first skip and the lift vector

points down. The vehicle stays in the atmosphere and completes its deceleration. Therefore, it has lower maximum deceleration values but also decreased flight range.

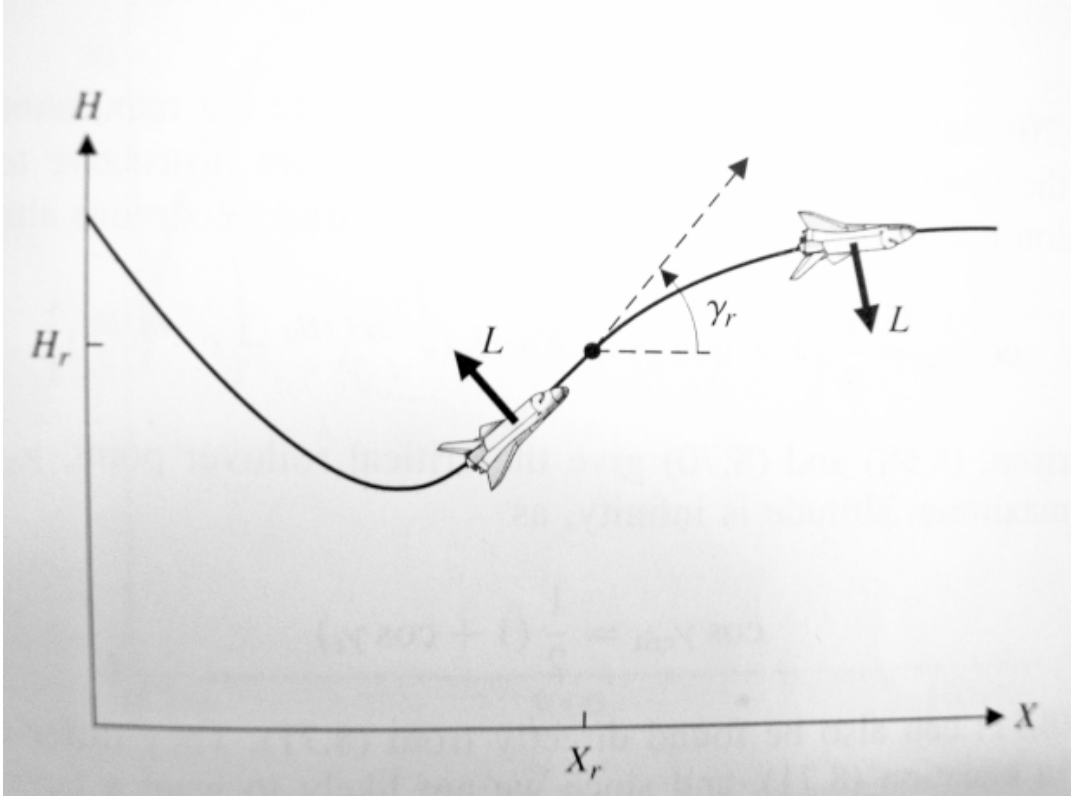

Figure 3. Double-Dip Entry [13]

Although the CEV will be capable of surviving more than 15g, which is considered the worst-case scenario during the reentry phase, CEV mission durations will be significantly longer than Apollo Program. This will subject astronauts to micro and low gravity for long periods and may require more constraining limits on deceleration to ensure the safety of physiologically deconditioned astronauts.

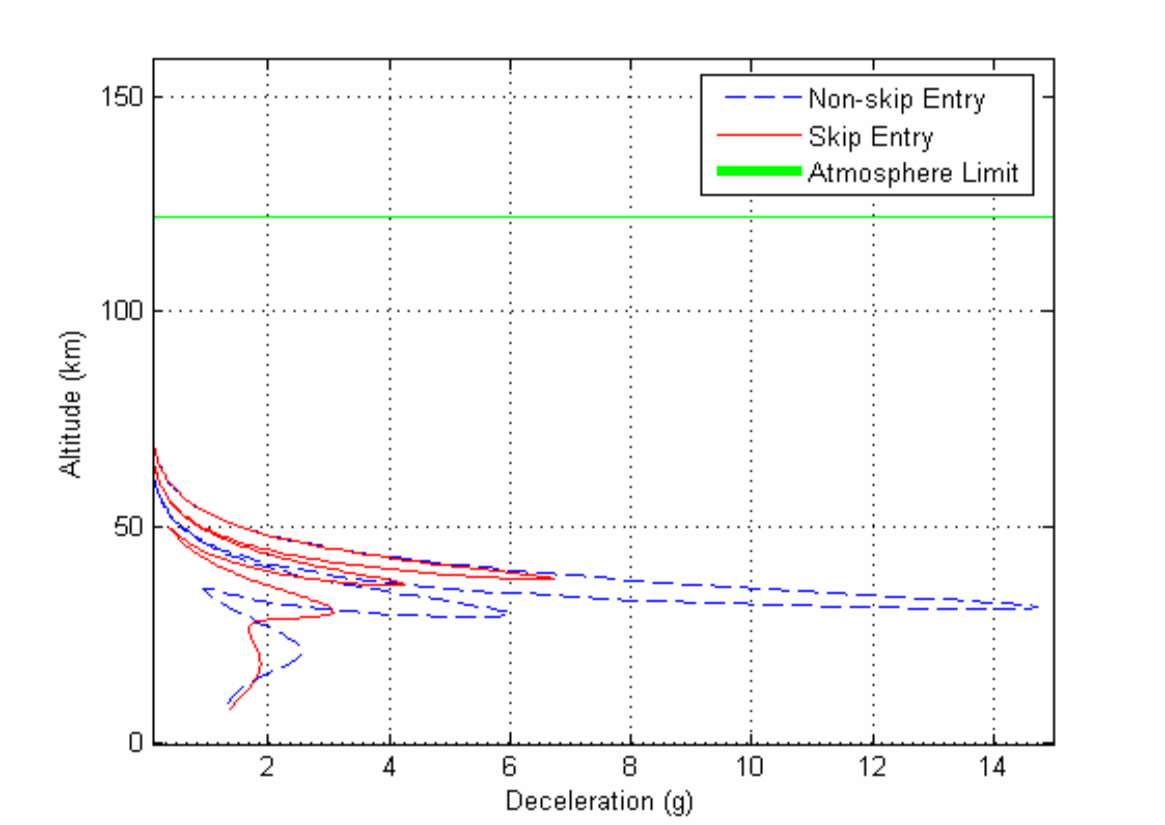

Figure 4. Skip and Non-skip Entry Trajectories (Altitude vs. Deceleration)

Performing a complete skip entry trajectory instead of Apollo's double dip entry trajectory will be more beneficial for the mission safety by giving the flexibility to choose the lunar return time. Astronauts will be exposed to lower deceleration rates and vehicle will be able land precisely on the predetermined landing sites.

Problem Statement

The CEV will be able to have both ground and water landing capability; however, it is considerably safer to land at the predetermined landing sites for the immediate ground support and recovery of the vehicle and astronauts. Having a flexible take off

time from the moon requires the calculations of spontaneous entry parameters that will be done by the CEV flight computer. Early determination of the reentry parameters will have significant effect on a successful lunar return and, it will help save the propellant that could be used for maneuvering and attitude control if any unpredicted error occurs in the trajectory and during reentry.

In order to solve the landing site determination problem successfully, a time-based solution must be applied under some assumptions. Since the landing sites will be constantly changing their location according to the inertial frame with the rotation of the Earth, reentry flight distance will be constantly increasing or decreasing while on the lunar return trajectory and also during the reentry phase.

Early determination of the reentry location in geodetic coordinates also appears to be a problem since the Moon does not have an equatorial orbit around the earth and its orbit is tilted between 18.28-28.58 degrees [20], depending on its current position. Figure 5 is a non-scaled simulation of the tilt angle of the moon with respect to the Earth's equator. To be able to overcome this problem, a coordinate rotation has to be made for the calculation, expressing the tilt angles in Earth-Centered Earth-Fixed (ECEF) coordinates, as well as a reverse rotation for the results.

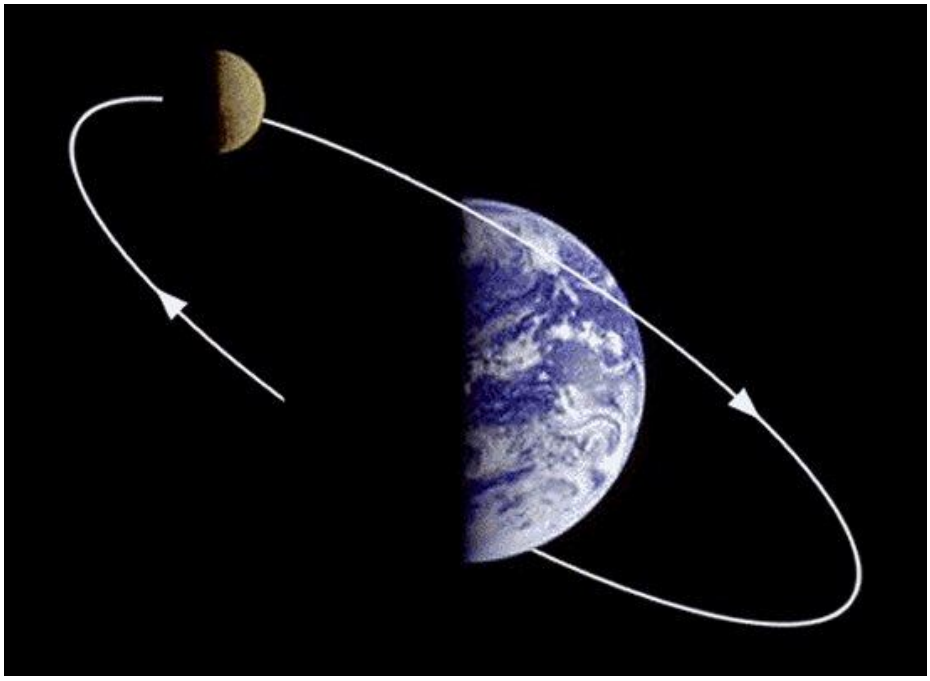

Figure 5. Illustration of the Moon around the Earth [18]

Like most manned and unmanned aerospace vehicles, deceleration effects are also significantly important for the CEV because of both structural and the human g tolerances. Structural limits are usually much higher than crew g limits, so that the reentry problem could be solved for two different entry options depending on the g limits of the current configuration, but increasing deceleration also means increasing energy dissipation and drag force on the structure. More drag on the structure also creates more heat on the heat shield. The heating rates are a concern because they impact the maximum instantaneous heat rejection rates. Tradeoffs between these two are often necessary. For example, long flights at high altitude reduce the heating rates but last longer so the total heating increases.[3] In a skipping reentry trajectory, flying out of the atmosphere has a large effect on cooling the vehicle down and getting it ready for the

next dip with lower kinetic energy. Therefore, a skipping reentry can be considered as the best option in a tradeoff decision for reentry with its beneficial effects and not very complicated nature. The animation in Figure 6 represents a good visual expression of the skip entry trajectory.

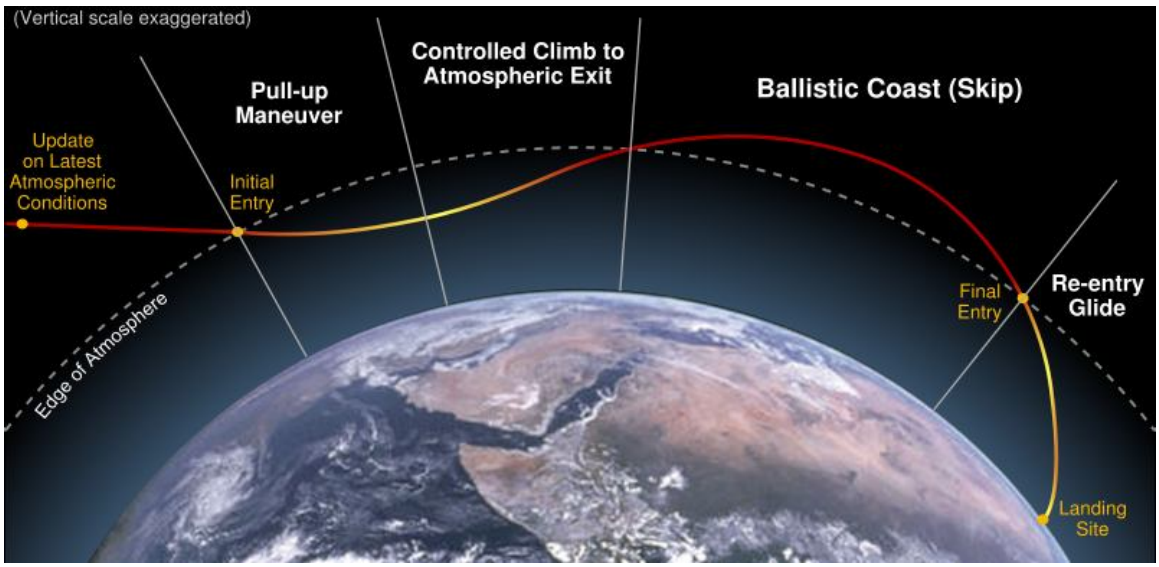

Figure 6. Skip Entry Trajectory [21]

Research Objectives

The main objective of this research effort is to establish that the lunar return can be initiated any time during a mission for emergency or mission completion purposes, and an accurate reentry can be accomplished at any landing site by adjusting the skip parameters. This procedure will be done by using an onboard reentry guidance computer than will process the reentry parameters for skip entry solution. Thus, the concept of entry time window will not be needed for accurate landing purposes.

On the other hand, the reentry guidance still relies on precise navigation information. This information can be a result of GPS data or any type of inertial navigation system located onboard. Any error in reentry coordinates or parameters will result a significant error in the landing location. Especially the results of entry coordinate errors will be hard predict due to the non-linear nature of the coordinate system.

II. Literature Review

Chapter Overview

The purpose of this chapter is to describe and analyze the previous research efforts in atmospheric reentry. It is well known that reentry is the most critical part of the overall return mission, and the reentry guidance algorithm plays an important role in steering the vehicle safely through the dispersed reentry flight environment, while meeting the mission requirements. There have been many research efforts on this topic and all tried to find the best feasible solution for the reentry problem of various vehicles including the space shuttle, Kistler K-l Orbital Vehicle, and Crew Exploration Vehicle (CEV), recently named Orion.

The academic papers and journals presented here are the different approaches to the reentry problem. Some different types of reentry techniques are considered for the CEV, including space shuttle type entry. The main idea of space shuttle type reentry from a lunar return trajectory is, firing the CEV engines to put the vehicle in a Low Earth Orbit (LEO). After that, the problem becomes a space shuttle type entry problem and will have about 16 entry windows in a 24 hour day period.

Other research examples are the references to this thesis work and act as guidance through the problem solution. Reading and studying previous works help to understand the topic better as well as showing different aspects to approach the problem.

Relevant Research

Space Shuttle Reentry Guidance

The Space Shuttle entry guidance provides steering commands to control the entry trajectory from initial penetration of the Earth's atmosphere (altitude of 122 km and range of approximately 7600 km from runway) until activation of the terminal area guidance. The terminal area guidance occurs at an Earth-relative speed of 762 m/s and at the that point, the shuttle is approximately 92 km from the runway threshold at an altitude of about 24 km. The primary objective of the entry guidance is to guide the shuttle along a path that minimizes the demands on the shuttle systems design and to deliver the vehicle to the best possible energy state and attitude at the initiation of the terminal area guidance system. The Space Shuttle entry guidance is designed to be able to analytically define a desired drag acceleration profile and command the vehicle to be at the right altitudes to achieve the desired reentry profile. This drag acceleration profile fits best to minimize the accumulated aerodynamic heat load throughout the entry corridor. [4]

The commanded Lift-to-Drag (L/D) value of the shuttle can be achieved by angle-of-attack modulation, by bank angle modulation, or by a combination of the two. The entry guidance of the shuttle uses a combination of bank angle and angle-of-attack modulation for trajectory control. Bank angle is the primary trajectory control parameter because the angle of attack can then be selected to minimize the aerodynamic heating environment while achieving the required cross range. In Figure 7, the bank angle profile of the shuttle throughout the trajectory can be seen in 50 cases simulated by the entry guidance system.

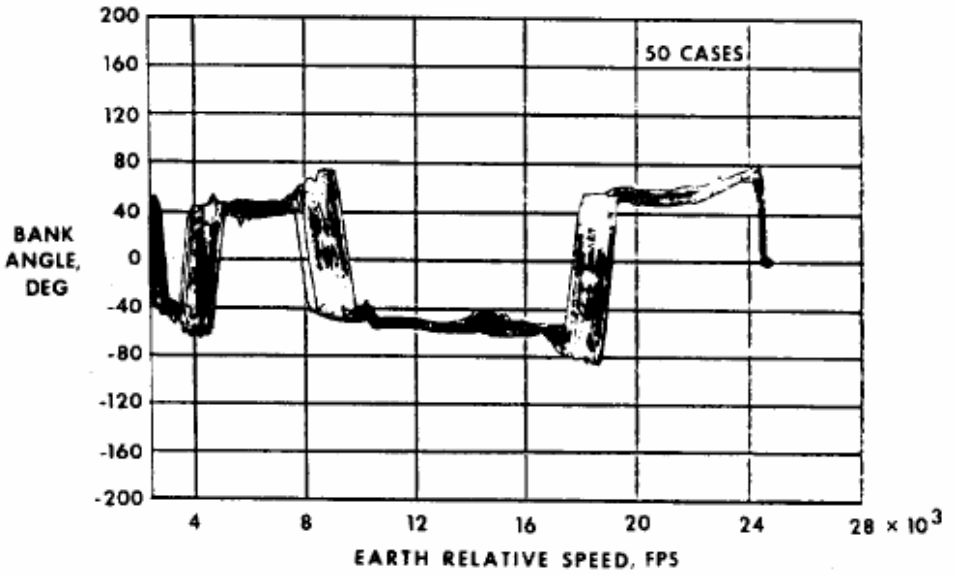

Figure 7. Bank Angle Profile of Space Shuttle [4]

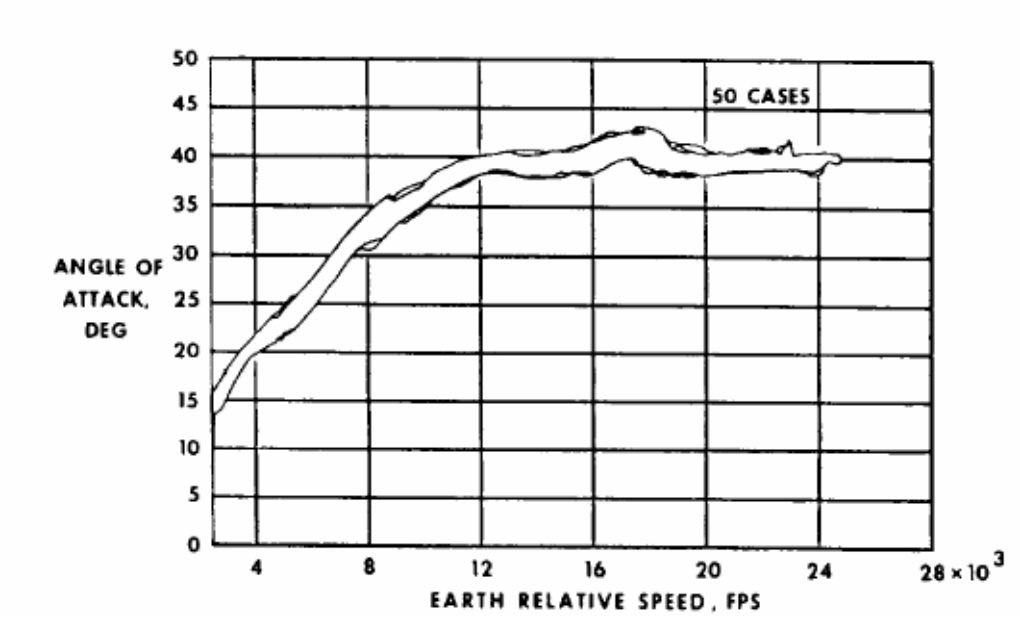

Figure 8. Angle of Attack Profile of Space Shuttle [4]

Figure 8 shows the changes in the angle of attack of the shuttle during reentry. These changes are made by the guidance system to achieve the best possible trajectory while keeping the vehicle under the maximum allowable g loads. This also minimizes changes in the aerodynamic heating distribution over the shuttle because of changes in the angle of attack. Therefore, bank angle is used to control both the total entry range and the cross range component of entry range.[4] The g load vs. speed graphic is shown in Figure 9.

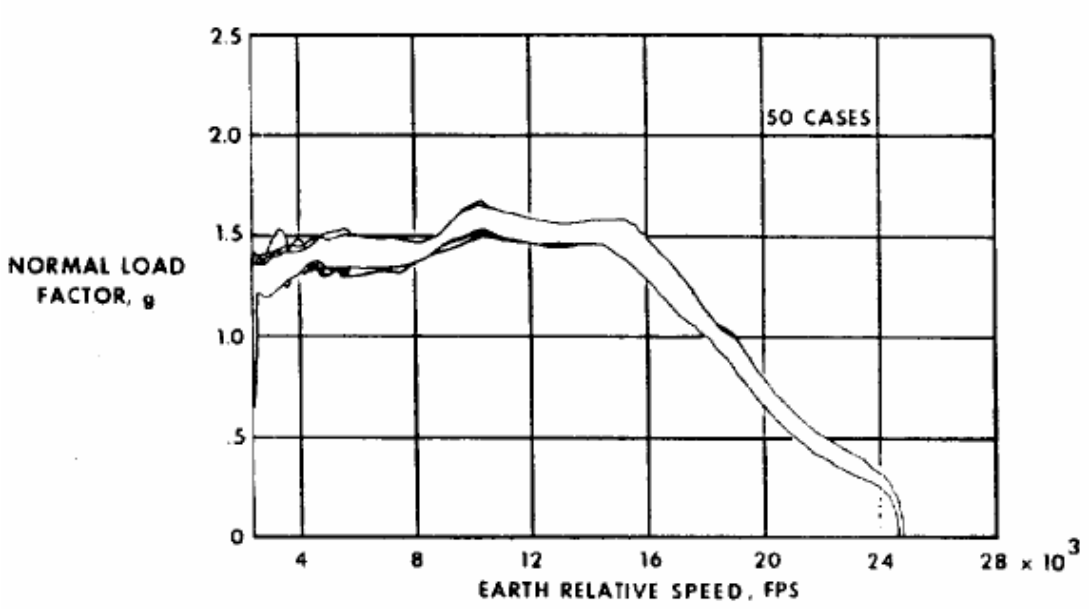

Figure 9. Normal Load Factor Profile of Space Shuttle [4]

GESARED Reentry Simulation

General Simulator for Atmospheric Reentry Dynamics (GESARED) is a simulation tool that was implemented in MATLAB® / SIMULINK. It was developed by the Delft University of Technology to provide an environment to design control laws for reentry vehicles. The simulation tool was meant to work on a personal computer.

GESARED was initially developed to design and test the guidance, navigation, and control (GN&C) systems for representative reentry vehicles. Its primary goal was to be the open-loop plant for reentry simulation where it can get the feedback from the GN&C algorithm at the latter point. Currently GESARED is the simulation environment used in the design of GN&C systems for the lifting body reentry vehicle (LBRV) and the atmospheric reentry capsule (ARC). The LBRV is a conceptual small reentry vehicle creating lift by flying high angles of attack. The vehicle has both side elevons and both side flaps as control surfaces. The ARC is an Apollo type guided and unmanned space capsule. It has successfully completed its first flight in 1998 including launch, suborbital ballistic flight, reentry and, descent. Because of the similarity in shape to the CEV, the ARC reentry experiment was an improved version of the original Apollo reentry algorithm, giving better results in terms of accuracy at landing. [5]

As seen in Figure 10, the ARC reentry trajectory is very close to the simulation data. It has an Apollo type reentry without using the double dip; however, it has a major difference at the reentry since it was not following a lunar return trajectory but a suborbital ballistic flight. Therefore, it has an entry velocity of 7.5 km/s as seen in Figure 11.

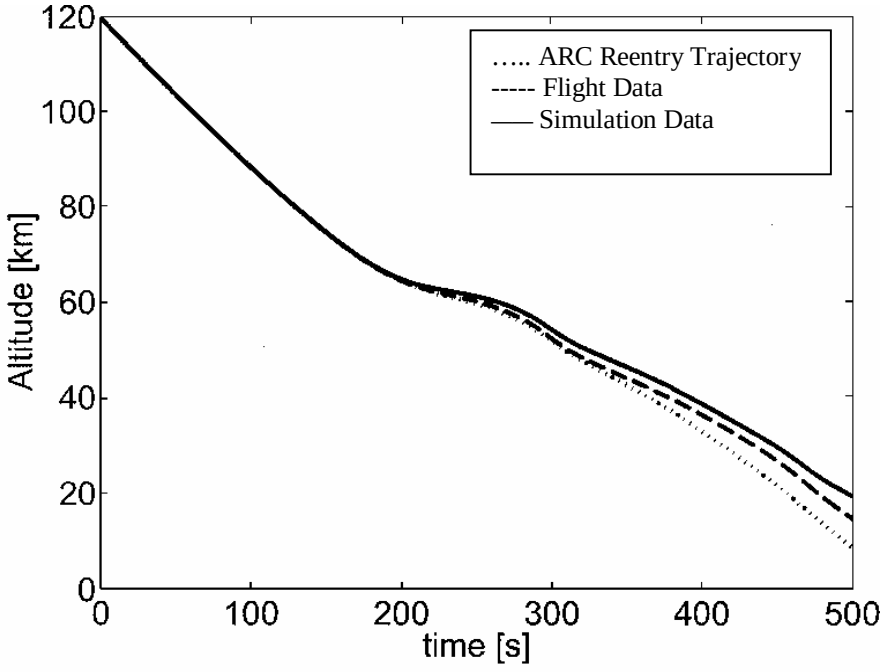

Figure 10. ARC Reentry Trajectory Altitude vs. Time [5]

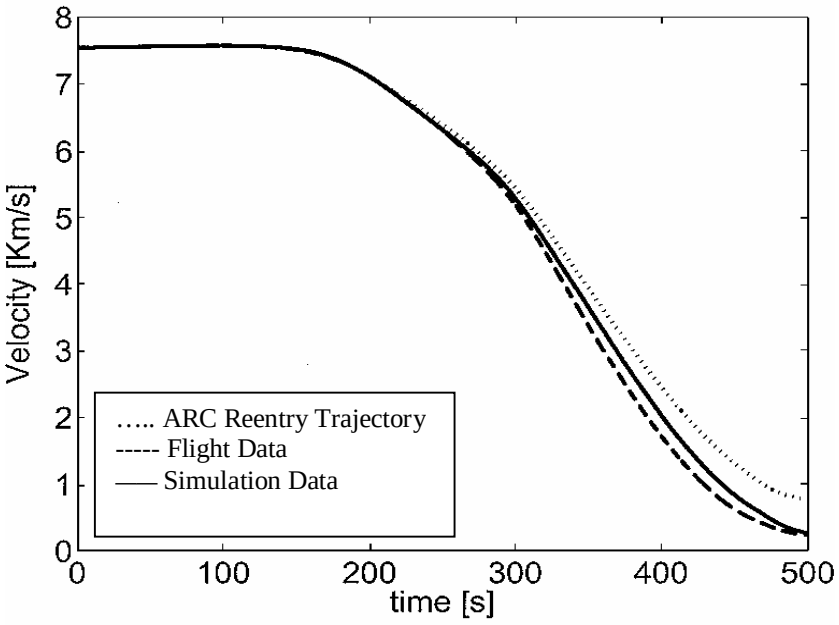

Figure 11. ARC Reentry Trajectory Velocity vs. Time [5]

Predictor-Corrector Reentry Guidance Algorithm

The main purpose of the Predictor-Corrector Reentry Guidance Algorithm is to focus on the evolution of the guidance strategy in order to satisfy both terminal and path constraints. During the each guidance cycle throughout the reentry trajectory, the program generates a feasible trajectory for the current conditions and compares it with the trajectory generated at the previous cycle. During this comparison, it also uses the measured flight data to make necessary changes on the current trajectory estimation. The predictor steering program uses the bank reversal philosophy as necessary to dissipate the vehicle's energy and reach the landing site. The path constraints include heat rate, aerodynamic load, and, dynamic pressure. These constraints are implemented as part of the algorithm to control the trajectory and adjust the control parameters within the allowable drag, and drag rate profiles. [6]

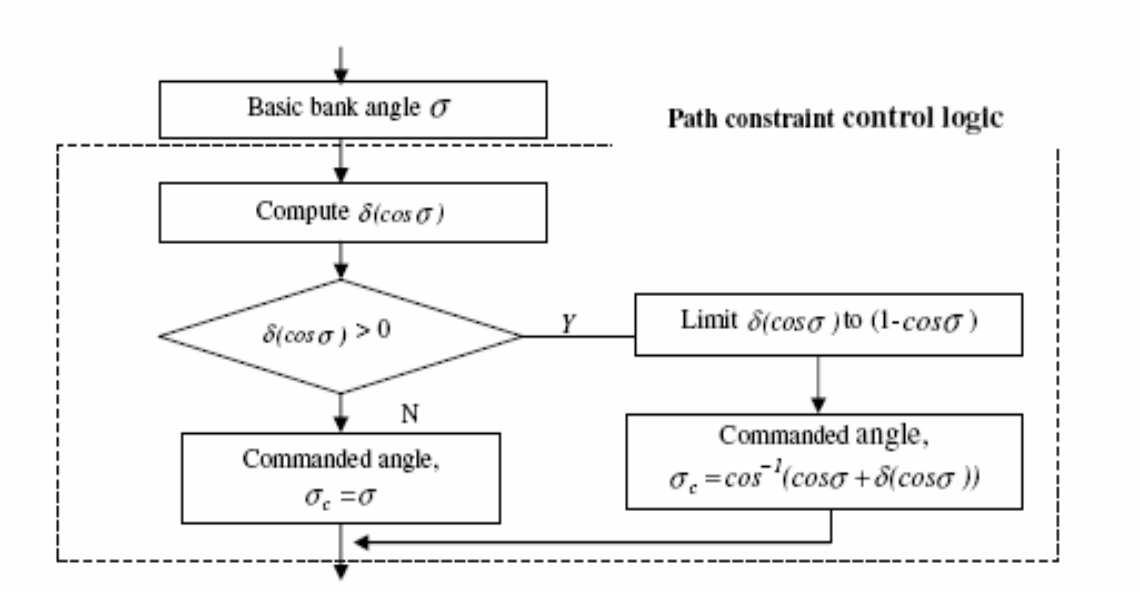

Figure 12. Schematic for Path Constraint Strategy [6]

In Figure 12, the bank angle modulation logic for trajectory control is shown. This logic is activated when the bank angle exceeds 10 degrees during the reentry. In each predictor step, it is ensured that the predicted trajectory satisfies the path constraints as seen in Figure 13.

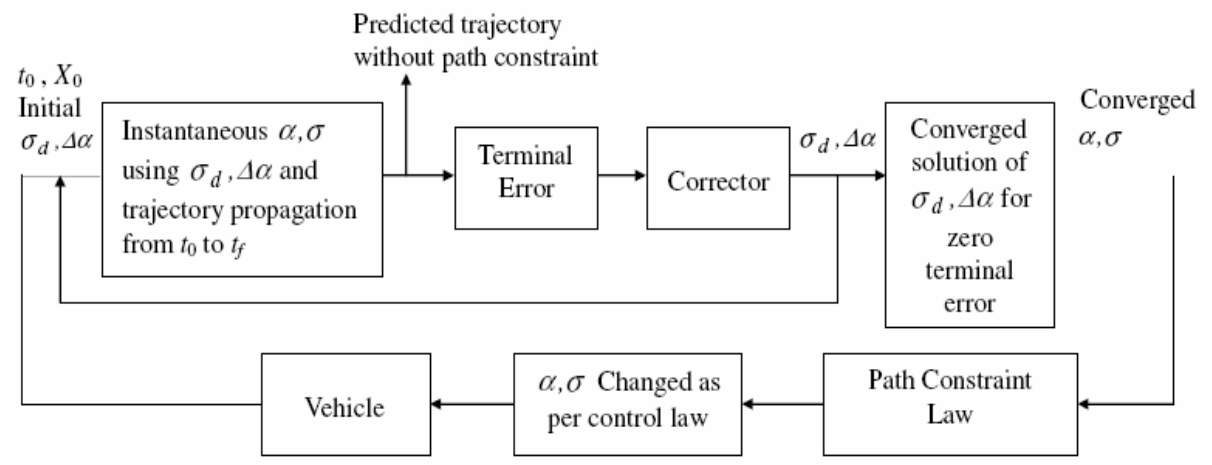

Figure 13. Path Constraint Activation at the Predictor-Corrector Output [6]

Figure 14 shows the results of the simulations for a typical reentry trajectory with the logic for heat rate constraint. The angle of attack (α) and bank angle (σ) are modulated to satisfy the path constraints in all guidance cycles, while the path constraint remains active. This process is repeated until the heat rate falls below the allowable limit. However, this changes the actual trajectory, which is then had to be adjusted by changing both α and σ during the later guidance cycles to meet the terminal constraints.

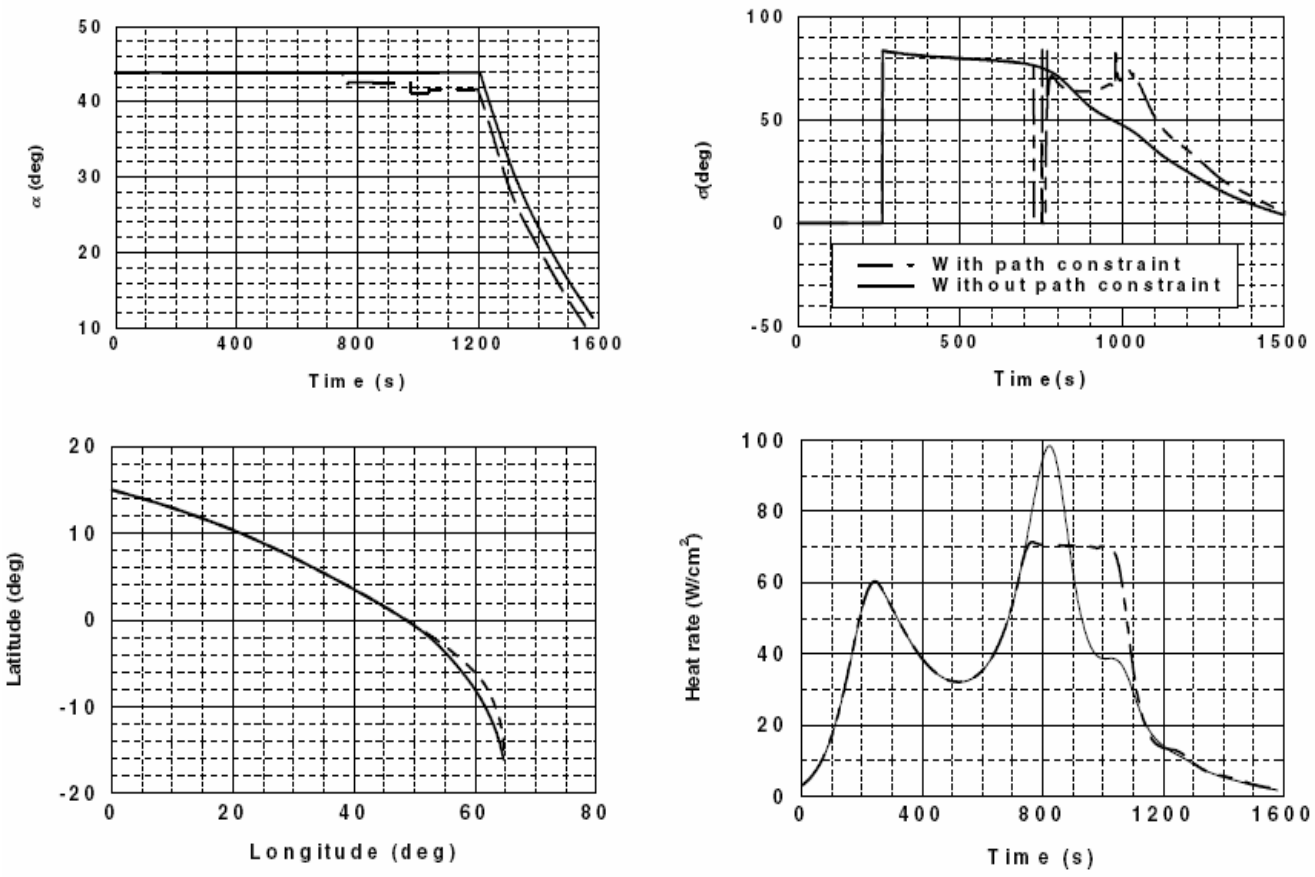

Figure 14. Predictor-Corrector Guidance Results with Path Constraint Control Strategy at the Algorithm Output Level [6]

Orion Reentry Guidance with Extended Range Capability

In this study, performance of the baseline Apollo algorithm was tested using a four degree-of freedom (4-DOF) simulation of the vehicle during reentry. Monte Carlo analyses were performed on this simulation in order to determine the results of the guidance algorithm in the presence of uncertainties. Then, two versions of the enhanced algorithm were developed and tested and the results were compared for consistency.

The vehicle used in the numerical simulation was assumed to have a constant mass throughout the trajectory, neglecting the loss of used fuel mass during reentry. A numerical simulation was implemented in MATLAB® version 7.0.4 in conjunction with Simulink version 6.2.

The atmospheric density model used in the simulation was the Standard U.S. Atmosphere, 1962. The lift and drag coefficients were taken as a function of altitude and Mach number, and the vehicle was assumed to be statically trimmed at all times. The fourth degree of freedom was the rotational motion of the vehicle described by the bank angle (φ) but the rotational torques which can affect the bank rate dynamics were not modeled. Instead, the bank angle of the vehicle was assumed to follow the closed-loop guidance bank angle commands. These commands were received at 2 second intervals and restricted by a 20 deg/sec rate limit. [7]

This enhanced guidance algorithm is based on the Apollo type reentry for the initial direct reentry part. However, the PredGuid program upgraded the phases relating to skip entry. These upgrades were sufficient to allow precise landing after skip entry for target ranges of up to 10,000 km. ground track. In Figure 15, it is seen that the CEP value in a 2400 km range test is 2.06 km. where this is under the required value of 3.5 km. [10]. The algorithm was quite robust even after giving some flight uncertainties and was successfully tested against certain stress cases. In addition, it was understood that the steepness of the skip can be controlled by modulating the time that the PredGuid takes over; starting earlier results in a steeper and higher altitude skip whereas starting

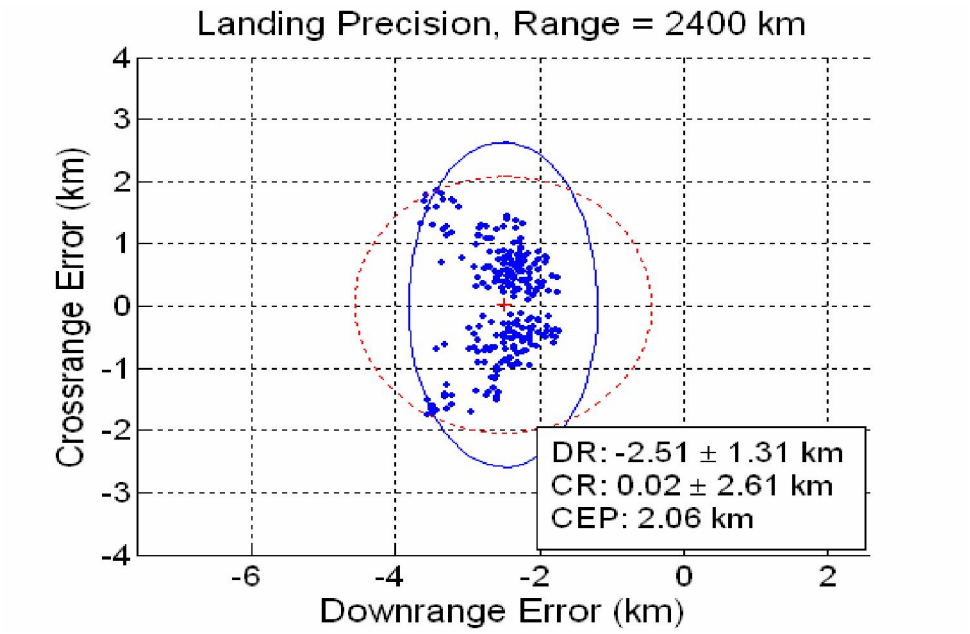

Figure 15. Typical Landing Error Distribution [7]

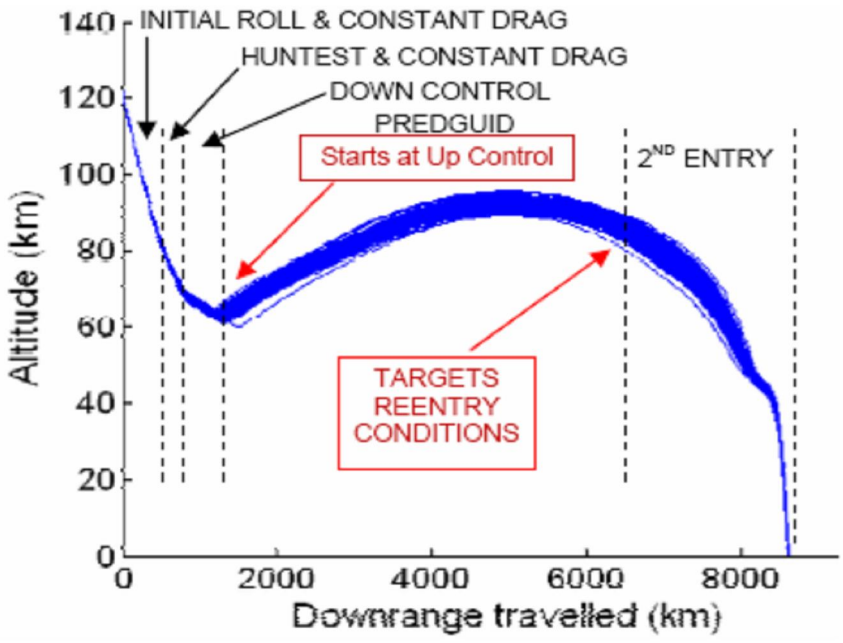

Figure 16. Enhanced PredGuid Algorithm [7]

later results in a shallower and lower altitude skip. Each of these options has its advantages and disadvantages. The change in the trajectory to achieve the target in 8400 km. range is seen after the PedGuid algorithm takes over the control in Figure 16. [7]

Trajectory Optimization for a Fixed-Trim Reentry Vehicle Using Direct Collocation and Nonlinear Programming

A fixed-trim reentry vehicle has negligible control over its angle-of-attack or sideslip angle and can only change its flight path by using its bank angle. Thus, the control variable is the vehicle bank angle for the rest of the reentry problem. There are also some other constraints that affect the solution such as the vehicle dynamics, initial and final conditions, and structural and thermal loading constraints. The specific vehicle in this work is the Kistler K-l Orbital Vehicle (OV). The OV is the second stage in a two-stage reusable launch system. The first stage Launch Assist Platform (LAP) lifts the vehicle to an altitude from which the OV can reach its orbit. After deploying a payload, the OV reenters the atmosphere and returns to the desired landing site.

In this study, the vehicle angle of attack and sideslip angle were assumed to remain at their trim values. In this case, reentry trajectory has two goals: minimizing the fuel used in attitude control system (ACS) and minimizing the deviation from the desired landing site. The collocation software is used to calculate the trajectory that results when the OV is held at a constant zero degree bank angle. The reentry simulation program starts working by receiving a desired bank angle command from the reentry guidance software. The control code estimates the current bank angle from the current vehicle

position, attitude, and velocity. Then, the software issues rotation commands in the vehicle roll and yaw axes. By doing this, the desired bank angle is maintained by the ACS jets. However, the controller has to constantly command to the jets to hold the bank angle inside the predetermined width.

In conclusion, the results showed that the final position error stayed below 1 nautical mile and the collocation software offered a significant savings in fuel. In addition to that, the g loads stayed under the constraints for all cases showing the collocation method is a feasible approach to solving the re-entry vehicle problem. [8]

.

A Comparison of Two Orion Skip Entry Guidance Algorithms

The two skip entry guidance algorithms that have been developed for the CEV are: the Numerical Skip Entry Guidance (NSEG) developed at NASA/JSC and PredGuid, developed at the Charles Stark Draper Laboratory.

Six degree-of-freedom analysis has been conducted with these two skip entry guidance algorithms. This analysis shows the feasibility of using a skip entry guidance algorithm to reach long-range targets up to 5,300 n.mi. from Entry Interface (EI) without using a correction maneuver out of the atmosphere. This skip entry range capability is thought to be able to access to the predetermined and alternate landing sites throughout the lunar month. There has been a performance comparison made by a senior selection board in order to select the primary and the alternate skip entry guidance algorithm after conducting several tests. The PredGuid algorithm was recommended as primary. The PredGuid algorithm demonstrated a better performance in Phase II in which a blended

bank angle command is used for the transition between the numerical solutions and the Apollo final phase solution. As a result, the NSEG algorithm will be kept as the backup algorithm and comparisons will be periodically performed to ensure that the optimum characteristics of both algorithms are identified and used the skip entry guidance algorithm. In Figure 17, it is seen that the PredGuid algorithm demonstrates a better trajectory solution in terms of accuracy until the drogue deployement compared to the NSEG algorithm. Since the flight path after the drogue deployment is not precisely controllable, the accuracy is evaluated until that time. [9]

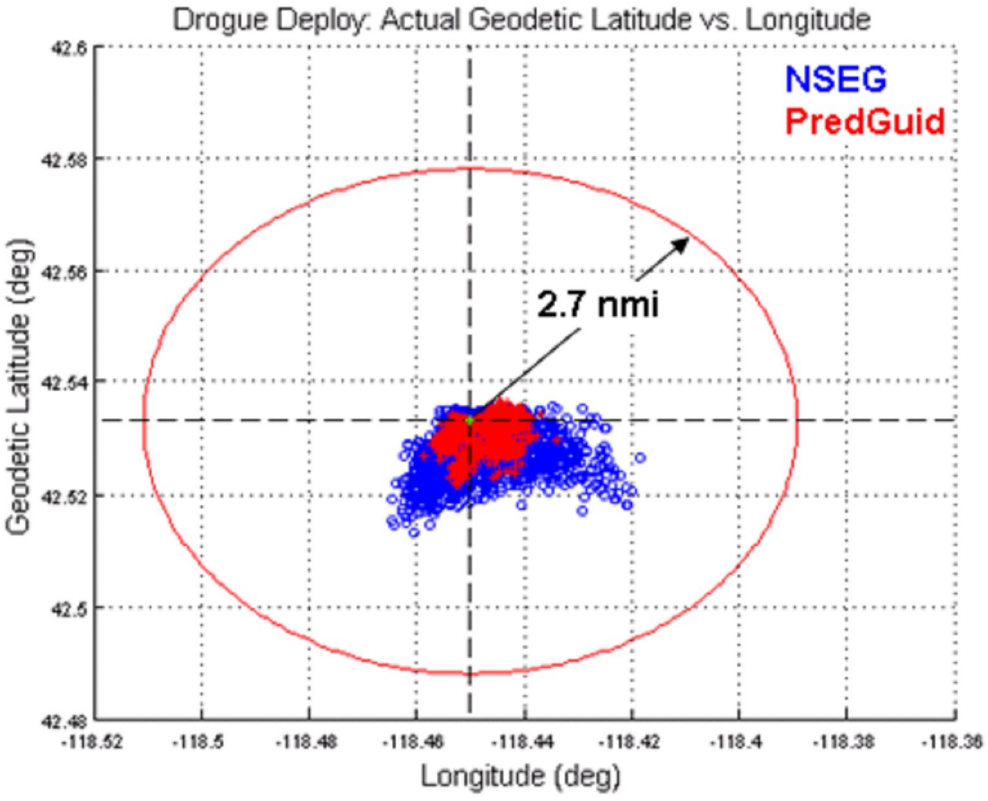

Figure 17. Guidance Algorithms Accuracy at Drag-Chute Deployment [9]

Summary

All of the researches presented in this section are the different perspectives to the same problem. The methods used in each are fairly close and most tried to find a solution using the post entry maneuvering of the CEV in the atmosphere by changing the bank angle and getting rid of their excess energy while staying on the predetermined trajectory. That also changed the ground track and became the one of the main sources of the errors. The research effort presented in this thesis will be a different approach to the reentry problem. The main purpose is to be able find the reentry conditions and parameters in order to have a steady state reentry trajectory unlike the ones presented here. There will be no major maneuvering within the atmosphere but the navigation system will still have to maneuver the vehicle slightly to take out the errors. This method of solution will provide the flexibility to initiate reentry whenever needed and having enough energy to land on the predetermined landing sites.

III. Methodology

Chapter Overview

The purpose of this chapter is to develop and explain the solution method for the CEV reentry problem. The approach to the problem and the solution will be described and the techniques and used formulas will be presented.

Problem Setup

As indicated previously, the reentry trajectory problem starts with the initiation of the lunar return procedures. The concentration of this research is to solve for the reentry parameters so that the reentry vehicle can keep a stable reentry throughout the trajectory. Therefore, as the return procedures start, the parameters have to be calculated depending on the time and position of the earth according to the moon. After solving for the parameters, the CEV will start its return trajectory to reach the calculated values and keep its attitude constant through the reentry phase. Calculated entry coordinates and flight path angle are going to be the key elements that are defining the whole trajectory within the atmosphere and during the skipping maneuver.

The skip-entry trajectory approach is not a new concept. The original Apollo guidance was developed with skip trajectory capability, which was never used because of navigation and control concerns during the skip maneuver. If the vehicle was skipped the atmosphere, it could have flown out above escape velocity, and could have never come back resulting a total catastrophe. In place of a total skip entry, Apollo used a double dip entry. The Soviet Union also used skip trajectories to return Zond robotic vehicles to a

Russian landing site. Considerable analysis was completed in the 1990s to investigate the long-range capability of vehicles in the 0.5 lift to drag ratio (C_l / C_d) class, which was considered the minimum L/D required to enable accurate skip trajectory entry capability at that time. [10]

The return trajectory begins with the targeting for the Trans-Earth Injection (TEI) maneuver while on the moon. The TEI maneuver is the propulsion maneuver used to set the CEV on a trajectory, which will intersect the Earth. The vehicle is placed on a trajectory that intercepts Entry Interface (EI) at 122 km. or 400,000 ft. at Earth at the correct flight path angle, latitude, longitude, and range to intercept the desired landing site. The flight path angle, reentry longitude, and latitude are controlled via the TEI maneuver during the departure of the moon. It establishes the required geometry to accomplish the return entry flight. The moon has a declination of maximum ± 28.6 deg. The entry vehicle enters the atmosphere at around 10.5 km/s. During the first dip, the flight path angle gradually increases. When the flight path angle is zero, the vehicle skips the first entry and its altitude starts increasing. During the coast to apogee, the navigation system is updated via GPS communication. Just before apogee of the skip orbit, a correction burn is executed using small engines on the capsule to correct for dispersions (if required) accumulated during the skip phase of the flight. This maneuver then helps the vehicle maintain the optimal set of reentry conditions at the second entry point. After executing the second entry with the right parameters, the vehicle targets for the landing site with no required bank angle change. A reference shape and basic dimensions of the CEV are shown in Figure 18.

Assumptions

The assumptions made in this research depend on the conceptual design of CEV model. Any type of change will also directly affect the results; however, the solution method remains valid.

The problem set up starts from the Moon for beginning of the solution. The Earth looks like a perfect giant ball from the moon. Although the moon is declined according to the Earth's equator, the perspective from the Moon's surface is a tilted, rotating sphere. Thus, the solution method presented here takes the Moon as a reference and the declination of the Moon orbit as Earth's tilt angle according to the reference. This tilt angle happened to be the first challenge during this research and solved by a coordinate rotation, which will be mentioned later.

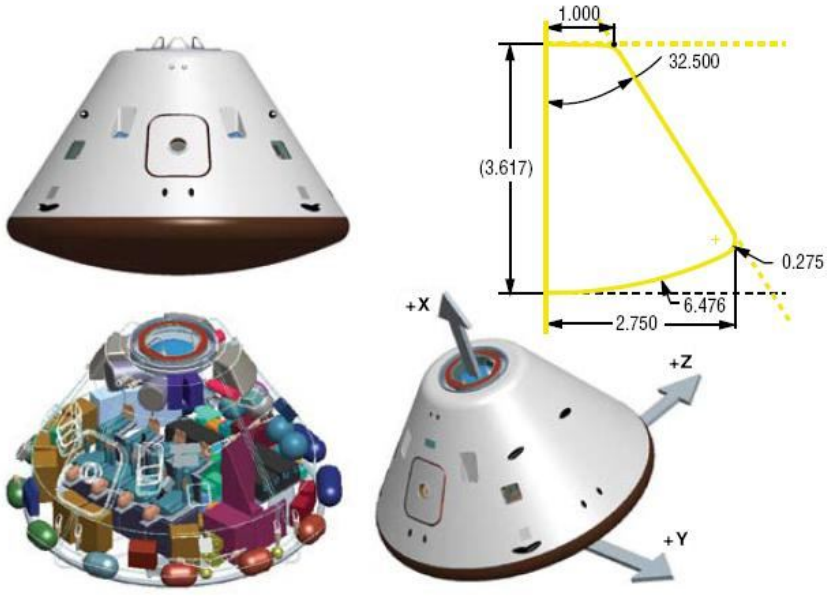

Figure 18. Apollo Derivative Crew Module [10]

The location of the landing sites are the other problem that has to be overcome. The rotation of the Earth causes the change of the locations according to the reference Moon surface. The rotation of the Earth is assumed constant. The time used in the solution method is based on the arrival time of the vehicle to the Earth's atmosphere. Since the return trajectory and duration between the Moon and Earth can be easily calculated after the departure, the atmosphere entry time can also be calculated.

The time calculation assumes the landing location is perfectly aligned with the moon departure location at time zero and the atmosphere entry time is expressed as the travel time of the landing location from time zero. For example; if the landing location is in the middle longitude of the other side of the Earth as viewed from the moon at the time of entry, then the entry time is assumed to be 12:00 since it was aligned with the departure location when the return began and now it is at the other side of the Earth, meaning 12 hours of rotation away. Sidereal time is not used in this study since the time is only a conceptual measure for the calculations; however, it could also be used with minor changes.

The vehicle properties such as the entry surface area, vehicle mass have different but similar values in different sources. The values in this study are taken from the NASA Exploration Systems Architecture Study (ESAS) Report. The vehicle mass is taken as 11500 kg. and the reentry surface area is taken as 23.76 m^2. Although it is not clear yet, the CEV is projected as an Apollo type capsule, which has a 0.4 lift to drag ratio (C_l / C_d) as shown in Figure 19.

Table 1. CEV General Parameters [10]

Lift Coefficient	0.443
Drag Coefficient	1.11
L/D	0.4
Aeroshell Diameter (m)	5.5
Mass (kg)	10,900
Ballistic Number (kg/m^2)	413.32

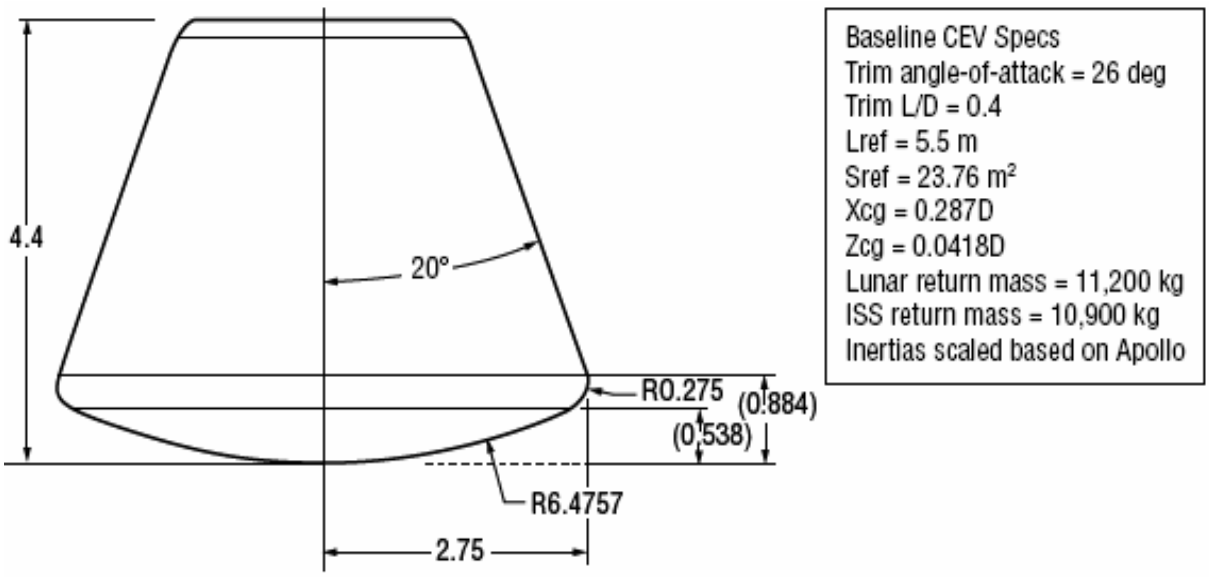

Figure 19. General Properties of CEV [10]

The entry coordinates are the second important parameters that have to be found for the solution. Since the landing location is described in the conceptual lunar departure time, then the reentry flight distance basically becomes the distance between entry point and landing location, which is the key parameter used in the solution method and will be

discussed later. However, the atmosphere is not perfect as it is assumed in the solution, the atmospheric effects such as weather events and high altitude winds are neglected to simplify the solution and a simple approximation for atmospheric density is used to form the "strictly exponential atmosphere," given by:

$$\rho = \rho_s e^{-\beta(r-R_\oplus)} \tag{3.1}$$

where ρ_s = atmospheric density at the surface, $R_\oplus$ = radius of Earth, β^{-1} = the scaling height that best matches the exponential atmospheric form. In addition to that, the lower layers of the atmosphere rotate with the rotation of the Earth decreasing as the altitude increases. Although this rotation rate is can be modeled and used in the solution, for simplification reasons, it is neglected in the solution method.

In addition to the assumptions made for the solution, one of the most important assumptions is considering the Earth as a perfect sphere. Although it doesn't make most of the calculations harder, the "unified theory" that is used for the solution of the problem works for a perfect spherical geometry. [3] On the other hand, the Earth can easily be considered as nearly perfect since its bulge is only about 0.33% of its radius.

Solution Method

Using the conceptual Moon departure time, as mentioned previously, the reentry time can be easily calculated and also the exact location of the landing locations can be found.

Entry Coordinates

The second problem is finding the entry coordinates. The entry coordinates are going to be the edges of the Earth. As seen in Figure 20, the picture seen from the Moon's perspective, where the return trajectory starts, the reentry points are shown in red and are unlimited. The aim of the return trajectory will be one of these reentry points with a flight path angle (γ) of slightly lower than zero, meaning the velocity vector ($\overrightarrow{{}^{R}V}$) pointing lower than the local horizon line as seen in Figure 21.

Figure 20. Representation of Possible Reentry Points [19]

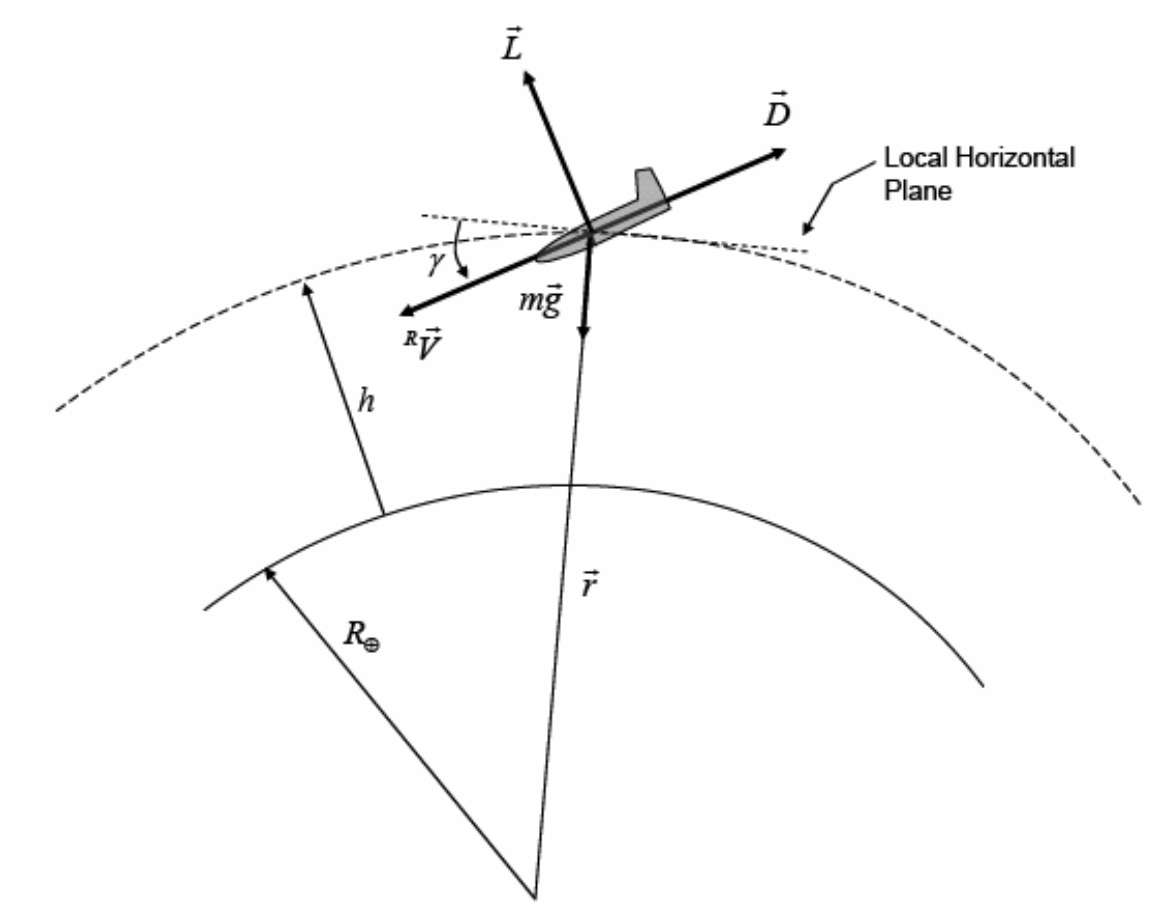

Figure 21. Two-Dimensional View of Planar Entry [3]

To be able to find the entry coordinates, the conceptual arrival time must be used. Since it defines the exact location of the landing site coordinates, the coordinates on the red line in Figure 21 can be found from there. For example, if the atmosphere arrival time is 2:00, that means the Earth rotated around 30 degrees. Let's say the landing site is at 280E – 28N coordinates (Kennedy Space Center). Remembering the assumption, made for the alignment of the landing site with the Moon at the departure time, the middle longitude will be 30 degrees less than the landing longitude. After finding the

mid longitude, the 2-longitude circle around the Earth can be found by adding and subtracting 90 degrees to the mid longitude. Therefore, in this case, the red line around the Earth will be composed of 160N and 340N longitudes. Either one of them can be selected for the reentry side but the selected side will define the entry type as either prograde or retrograde. Since the atmospheric activities and the motion of the lower layers of the atmosphere with the rotation of the Earth are neglected in this study, solving the problem for a prograde or a retrograde reentry type will only change the total reentry flight distance and therefore affecting the entry flight path angle (γ).

Finding the entry latitude can be done in a similar way. If the flight path of the vehicle is thought to be its orbit, the inclination of that orbit will give the entry latitude. In order to be able to find that inclination, the angle between the orbit plane and equatorial plane has to be found. This is a simple solution using the spherical trigonometry. As seen in Figure 22, the angle between a and c is equal to the angle between the OAC and BAC planes. Thus, by converging the C point to the intersection of mid longitude and zero degrees latitude (equator), point A to the location of the landing site and point B to the pole, it is now easy to get the inclination angle from the spherical trigonometry formulas.

$$\sin a = \sin c \cdot \sin \alpha \tag{3.2}$$

$$\cos c = \cos a \cdot \cos b \tag{3.3}$$

$$\sin b = \tan a \cdot \cot \alpha \tag{3.4}$$

$$\cos \beta = \tan a \cdot \cot c \tag{3.5}$$

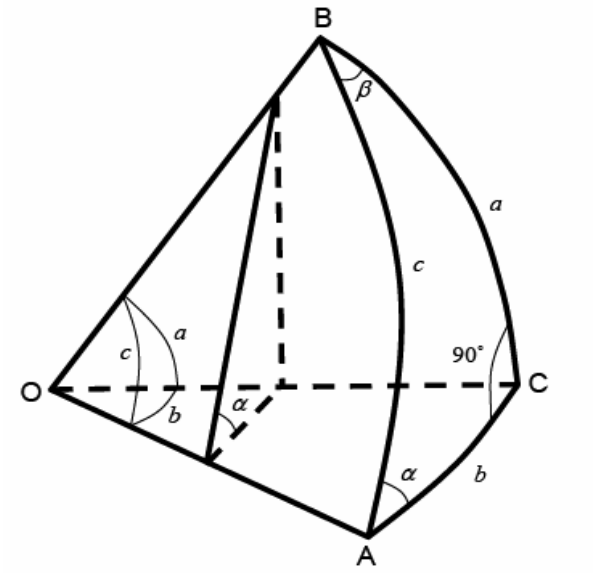

Figure 22. General Right Spherical Triangle [3]

Flight Distance

The distance mentioned as flight distance is actually the angular distance of the ground track between the EI point and landing site. Thus, the flight distance can also be calculated by using the spherical distance formulas.

$$\Delta L = \left|\theta_e - \theta_L\right| \tag{3.6}$$

$$\cos s = \sin\phi_L \cdot \sin\phi_e + \cos\phi_L \cdot \cos\phi_e \cdot \cos\Delta L \tag{3.7}$$

From these equations we can find the angular distance as:

$$s = a\cos(\sin\phi_L \cdot \sin\phi_e + \cos\phi_L \cdot \cos\phi_e \cdot \cos(\left|\theta_e - \theta_L\right|)) \tag{3.8}$$

The angular distance (s) gives the ground track of the trajectory, which later can be used to solve the equations related to the reentry.

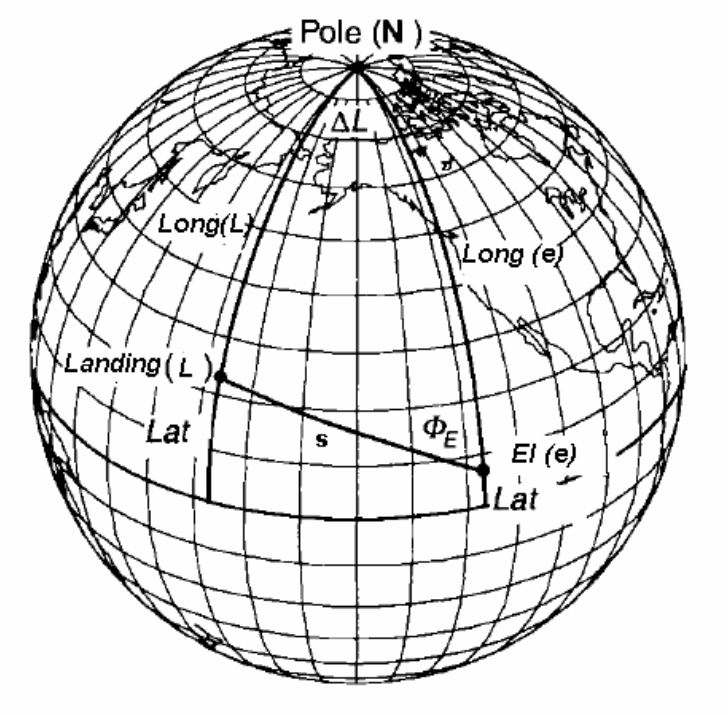

Figure 23. Relationship Between Landing Site and EI [15]

Coordinate Rotations

As previously mentioned, the declination of the Moon has a negative effect on projecting the entry coordinates. However, this problem can be overcome by doing a simple coordinate rotation according to the tilt angle seen from the Moon's perspective. Since the Earth is considered as a perfect sphere and the ground track of the flight distance is taken as a constant after the entry time calculations, the rotation made in the Geodetic coordinates will not affect the result. If the landing and the entry coordinates are rotated according to the tilt angle of the Earth, the problem can be solved with the

newfound pseudo coordinates. However, a reverse rotation of the coordinates has to be made at the end of the solution to present the entry and landing coordinates correctly.

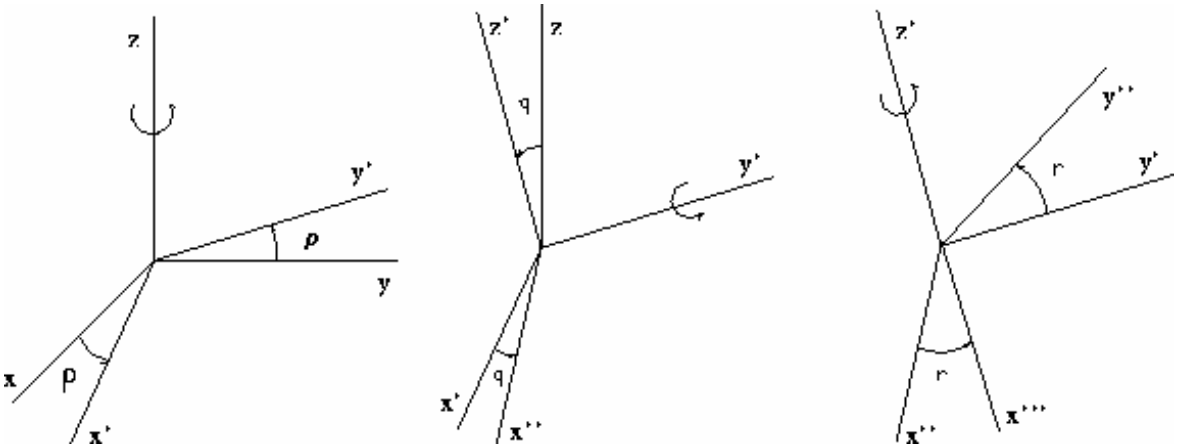

Figure 24. Coordinate Rotations [15]

The coordinate rotations are made as the angles are measured in a counter-clockwise direction and the following rotation matrices are used.

$$R_1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos p & -\sin p \\ 0 & \sin p & \cos p \end{pmatrix} \tag{3.9}$$

$$R_2 = \begin{pmatrix} \cos q & 0 & \sin q \\ 0 & 1 & 0 \\ -\sin q & 0 & \cos q \end{pmatrix} \tag{3.10}$$

$$R_3 = \begin{pmatrix} \cos r & -\sin r & 0 \\ \sin r & \cos r & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{3.11}$$

The rotation matrix is formed by: $R_1 \cdot R_2 \cdot R_3$

To be able make the coordinate rotation, the geodetic coordinates must be converted in to ECEF coordinates in the vector format. Next, the location vectors will be multiplied by the rotation matrix and the result will be converted back to the geodetic coordinate system. The conversion is made using these formulas: [17]

$$x = (N + h)\cos\phi\cos\theta \tag{3.12}$$

$$y = (N + h)\cos\phi\sin\theta \tag{3.13}$$

$$z = [N(1 - e^2) + h]\sin\phi \tag{3.14}$$

where:

$\phi, \theta, h =$ geodetic latitude, longitude, and height above ellipsoid.

$x, y, z =$ Earth Centered Earth Fixed Cartesian Coordinates, and;

$$N(\phi) = a / \sqrt{1 - e^2 \sin^2 \phi} \tag{3.15}$$

N= Radius of the curvature in prime vertical

a= semi-major Earth axis (ellipsoid equatorial radius)

b= semi-minor Earth axis (ellipsoid polar radius)

$$f = \frac{a - b}{a}$$
$$e^2 = 2f - f^2 \tag{3.16}$$

f= flattening

e= eccentricity

Reverse conversion from ECEF coordinates to geodetic coordinates are made by using these formulas: [17]

$$\phi = a\tan(\frac{z + e'^2 b\sin^3\lambda}{p - e^2 a\cos^3\lambda}) \tag{3.17}$$

$$\theta = a\tan 2(y, x) \tag{3.18}$$

$$h = \frac{p}{\cos\phi} - N(\phi) \tag{3.19}$$

where;

$$\begin{aligned} p &= \sqrt{x^2 + y^2} \\ \lambda &= a\tan(\frac{z \cdot a}{p \cdot b}) \\ e'^2 &= \frac{a^2 - b^2}{b^2} \end{aligned} \tag{3.20}$$

Unified Theory

In order to solve for the reentry problem the universal equations derived by Vinh and Brace are used.[11] These equations are independent of mass, size, and vehicle shape.

$$\frac{dZ}{ds} = -\overline{\beta r}Z\tan\gamma \tag{3.21}$$

$$\frac{du}{ds} = -\frac{2Zu\sqrt{\overline{\beta r}}}{\cos\gamma}\left(1 + \frac{C_L}{C_D}\cos\sigma\tan\gamma + \frac{\sin\gamma}{2Z\sqrt{\overline{\beta r}}}\right) \tag{3.22}$$

$$\frac{d\theta}{ds} = \frac{\cos\psi}{\cos\phi} \tag{3.23}$$

$$\frac{d\phi}{ds} = \sin\psi \tag{3.24}$$

$$\frac{d\gamma}{ds} = \frac{Z\sqrt{\beta r}}{\cos\gamma}\left[\frac{C_L}{C_D}\cos\sigma - \frac{\cos\gamma}{Z\sqrt{\beta r}}\left(1 - \frac{\cos^2\gamma}{u}\right)\right] \tag{3.25}$$

$$\frac{d\psi}{ds} = \frac{Z\sqrt{\beta r}}{\cos^2\gamma}\left(\frac{C_L}{C_D}\sin\sigma - \frac{\cos^2\gamma}{Z\sqrt{\beta r}}\cos\psi\tan\phi\right) \tag{3.26}$$

In addition to these six equations, using the Vinh's equation that is used to change the independent variable from time to "s," the time solution for the reentry can be extracted.

$$s = \int_0^t \frac{{}^R V}{r}\cos\gamma\, dt \tag{3.27}$$

Using Eq.(3.18), if the time is extracted:

$$\frac{dt}{ds} = \frac{r}{{}^R V\cos\gamma} \tag{3.28}$$

To be able to use this equation, the dependent variables has to be exchanged with the independent ones. Using the Vinh's dependent variable change equations:

$$u = \frac{{}^R V^2\cos^2\gamma}{gr} \tag{3.29}$$

$$Z = \frac{\rho C_D S}{2m}\sqrt{\frac{r}{\beta}} \tag{3.30}$$

and the gravity term:

$$g = g(r) = \frac{\mu}{r^2} \tag{3.31}$$

the time solution equation turns out to be:

$$\frac{dt}{ds} = \frac{r\sqrt{r}}{\sqrt{u\mu}} \tag{3.32}$$

After the variable changes are made, the ”non-dimensional altitude variable” [3] has to be found using:

$$\eta = \frac{\rho S C_D}{2m\beta} \tag{3.33}$$

Then, the altitude variable becomes:

$$\eta = Z / \sqrt{\overline{\beta r}} \tag{3.34}$$

If the $\overline{\beta r}$ is assumed to stay almost constant throughout the trajectory then it can be replaced with a constant. Now altitude can easily be found using η. This assumption is consistent with Unified theory since the equations in Unified Theory were found using the same assumption.

For a MATLAB® solution of these 7 equations, some variable changes has to be done:

$X_1 = Z$
$X_2 = u$
$X_3 = \theta$
$X_4 = \phi$
$X_5 = \gamma$
$X_6 = \varphi$
$X_7 = t$

Using the new variables, the unified theory equations can be rewritten with the time solution:

$$\dot{X}_1 = -\beta r X_1 \tan(X_5) \tag{3.35}$$

$$\dot{X}_2 = \frac{-2X_1X_2\sqrt{\beta r}}{\cos(X_5)}\left(1 + \frac{C_L}{C_D}\cos(\sigma)\tan(X_5) + \frac{\sin(X_5)}{2X_1\sqrt{\overline{\beta r}}}\right) \tag{3.36}$$

$$\dot{X}_3 = \frac{\cos(X_6)}{\cos(X_4)} \tag{3.37}$$

$$\dot{X}_4 = \sin(X_6) \tag{3.38}$$

$$\dot{X}_5 = \frac{X_1\sqrt{\beta r}}{\cos(X_5)}\left(\frac{C_L}{C_D}\cos(\sigma) + \frac{\cos(X_5)}{X_1\sqrt{\beta r}}\left(1 - \frac{\cos^2(X_5)}{X_2}\right)\right) \tag{3.39}$$

$$\dot{X}_6 = \frac{X_1\sqrt{\beta r}}{\cos^2(X_5)}\left(\frac{C_L}{C_D}\sin(\sigma) - \frac{\cos^2(X_5)}{X_1\sqrt{\beta r}}\cos(X_6)\tan(X_4)\right) \tag{3.40}$$

$$\dot{X}_7 = \frac{r\sqrt{r}}{\sqrt{X_2\mu}} \tag{3.41}$$

Since MATLAB® ODE function works in the matrix form, the equations have to be turned into matrix form:

$$\begin{bmatrix} \dot{X}_1 \\ \dot{X}_2 \\ \dot{X}_3 \\ \dot{X}_4 \\ \dot{X}_5 \\ \dot{X}_6 \\ \dot{X}_7 \end{bmatrix} = \begin{bmatrix} -\beta r\tan(X_5) & 0 & 0 & 0 & 0 & 0 \\ \frac{-2X_2\sqrt{\beta r}}{\cos(X_5)}\left(1 + \frac{C_L}{C_D}\cos(\sigma)\tan(X_5)\right) & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{X_1\sqrt{\beta r}}{\cos(X_5)}\frac{C_L}{C_D}\cos(\sigma) & 0 & 0 & 0 & 0 & 0 \\ \frac{X_1\sqrt{\beta r}}{\cos^2(X_5)}\frac{C_L}{C_D}\sin(\sigma) & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \\ X_5 \\ X_6 \\ X_7 \end{bmatrix} + \begin{bmatrix} 0 \\ -X_2\tan(X_5) \\ \frac{\cos(X_6)}{\cos(X_4)} \\ \sin(X_6) \\ \left(1 - \frac{\cos^2(X_5)}{X_2}\right) \\ -\cos(X_6)\tan(X_4) \\ \frac{r\sqrt{r}}{\sqrt{X_2\mu}} \end{bmatrix} \tag{3.42}$$

where $\left[\dot{X}\right] = A[X] + B$

Flight path angle (γ), heading (ψ), latitude (θ), longitude (ϕ), and time (t) will be the direct results of these inputs. However, the altitude (h) and velocity (v) have to be extracted using Vinh's equations in reverse.

If $\overline{\beta r}$ is assumed to stay constant throughout the trajectory, then it becomes easy to calculate the altitude from η. Using Eqs. (3.1), (3.30), and (3.34), the altitude is becomes:

$$h = \frac{1}{-\beta \ln\left(\frac{2m\eta\beta}{S\rho_s C_D}\right)} \tag{3.43}$$

The scalar velocity of the vehicle can also be extracted using Eqs. (3.29) and (3.31), and becomes:

$$v = \frac{\sqrt{\frac{u\mu}{r + R_\oplus}}}{\cos\gamma} \tag{3.44}$$

Deceleration and Heating Calculation

Deceleration on the vehicle is a function of the drag force acting on it during reentry. As it is seen on the Figure 25, most of the deceleration occurs at the altitude of around 40 km. altitude. The main reason for this is the exponentially increasing atmospheric density function. Some of the examples of atmospheric densities according to the altitude changes are presented in Table 2.

Table 2. Altitude Air Density Relationships

Altitude (km):	50	45	40	35	30
Density(kg/m^3):	1.117e-003	2.249e-003	4.529e-003	9.122e-003	18.37e-003

As it is seen in Table 2, the atmospheric density change in the lower altitudes are enormous, causing most of the drag on the reentry vehicle and dissipating its energy. However, most of its energy is dissipated between the altitudes 40 and 45 km. and the deceleration rate decreases even though the atmospheric density is doubled in the lower

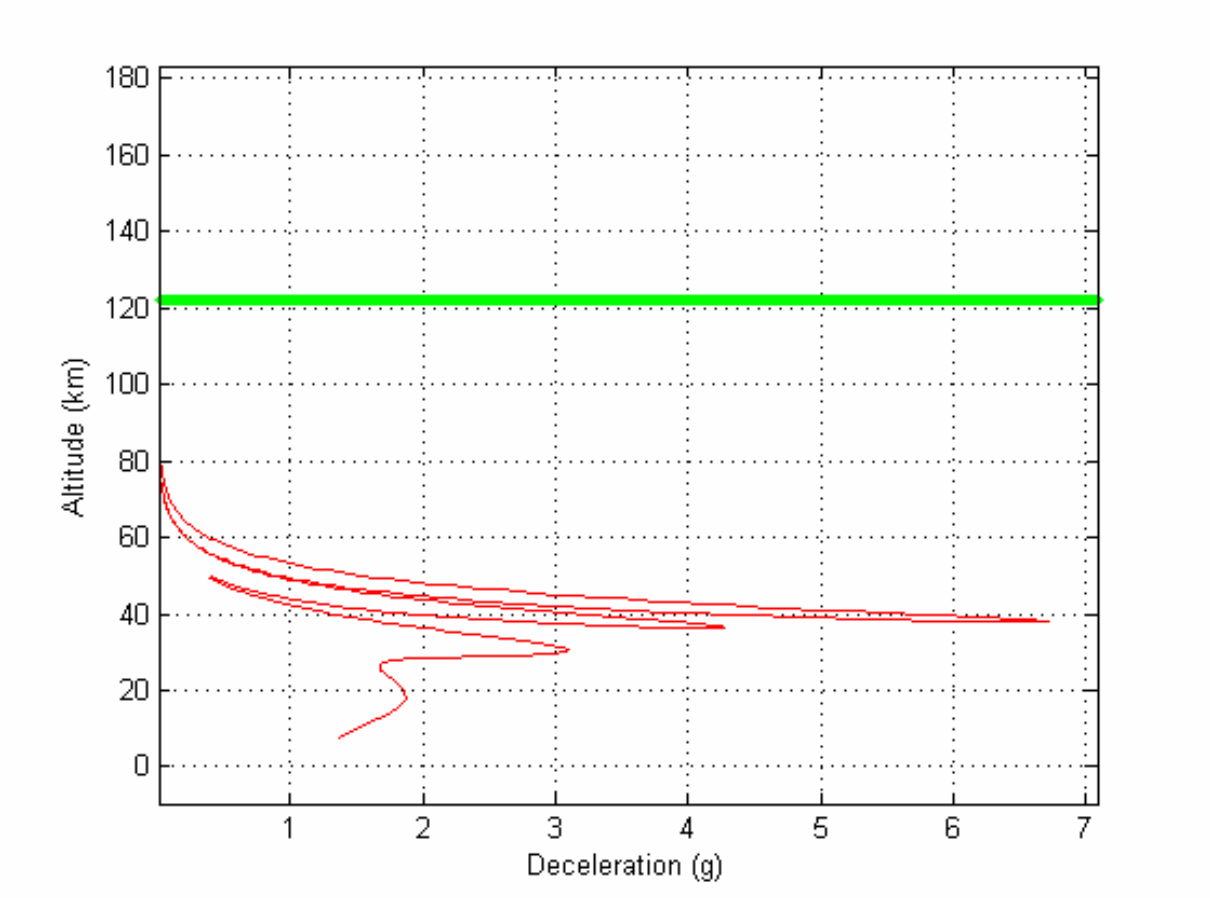

Figure 25. Deceleration vs. Altitude

altitudes. The deceleration on the vehicle is found using Loh's second order solution for deceleration. [12]

$$\frac{a_{decel}}{g_0} = 2\beta r_0 \eta T \sqrt{1 + \left(\frac{C_L}{C_D}\right)^2} \tag{3.45}$$

In order to solve the deceleration equation the kinetic energy of the vehicle must be found. After redefining the kinetic energy in terms of universal equations, it becomes: [3]

$$T = \frac{u}{2\cos^2 \gamma} \tag{3.46}$$

Using kinetic energy, now the stagnation and wall heat flux parameters can be found using:

$$\dot{q}_w = \eta T^{3/2} \tag{3.47}$$

$$\dot{q}_s = \eta^{1/2} T^{3/2} \tag{3.48}$$

Unsurprisingly, the wall heat flux and stagnation heat flux versus altitude graphics look very similar to the deceleration graphic. The main reason for this is the kinetic energy parameter in all three equations. In this example, the kinetic energy of the vehicle decreases very fast around the altitudes 35-45 km. because of the increasing drag force with the increasing air density. However, there is a unique difference between the deceleration and heat flux graphics. The peak values on the heat flux are achieved at 33 km. but the peak deceleration rate is achieved at 38 km. altitude, and the air density is the

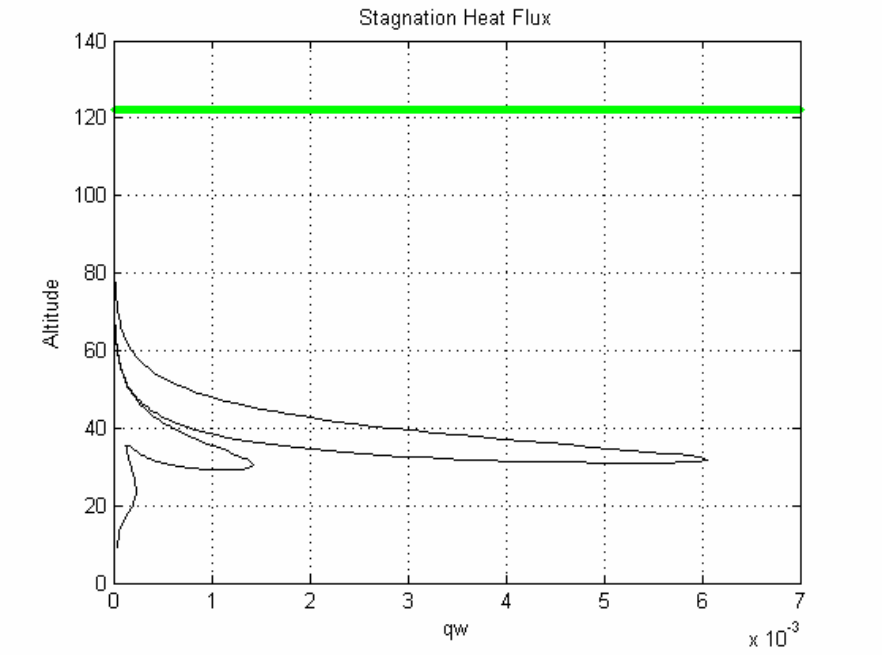

Figure 26. Wall Heat Flux vs. Altitude

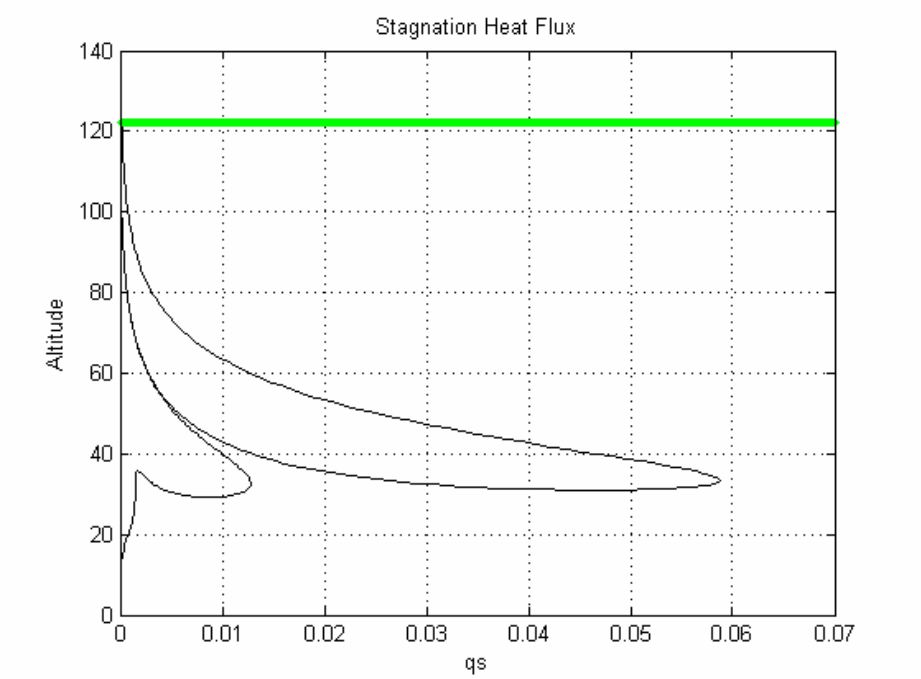

Figure 27. Stagnation Heat Flux vs. Altitude

same exponential increasing model in both subjects. The reason for this event can be explained as the heating does not occur as quickly as the deceleration since the deceleration is a result of sudden increased drag force. The heat flux also occurs because of the drag force but it has a cumulative nature. Therefore, it starts building up with an increased rate at the same altitude with peak deceleration, but the peak heat flux is achieved when it comes to equilibrium with the surrounding air and then goes down as the kinetic energy and drag force decreases. Thus, maximum the heat flux is expected to happen after the peak deceleration rate as experimented in the example.

Summary

In this section, the solution method for the reentry problem is presented. The main idea for the solution method was to simplify the entry and achieve an accurate landing on the predetermined landing site. It is considered that this solution has two different benefits for the overall mission. The first and probably the most important benefit is being able to get rid of the reentry window concept in order to make accurate landings. Since more than two pi radians of angular distance can be obtained by changing the entry flight path angle and completing a full skip entry trajectory, this concept gives the eligibility to access any landing site on Earth. As it is mentioned in the literature review part, the reentry trajectories mostly deal with a constant or very little changing flight path angles and define their atmospheric trajectories and flight paths by changing the bank angle of the vehicle for energy dissipation and also navigation purposes. Thus, these types of reentry models require an entry time window for the vehicle, which is normal for the normal procedures, but causing problems in case of an

emergency demanding an earlier or later return. The other benefit is the simplicity of the trajectory. The bank angle of the vehicle is kept constant through out the trajectory and the entry parameters are calculated at the very beginning of the lunar departure. Although the weather effects atmospheric movement are not considered and involved in the calculations, they can still be compensated in the skip part or in the atmosphere.

IV. Analysis and Results

Chapter Overview

This chapter includes a brief description of the software that is developed in order to solve the reentry problem using MATLAB® , and the analysis of the program structure and the results will be presented. First, user operations on the program will be described and, next, the design of the algorithm and the processes will be outlined. The program functions and the problem solution will be displayed with the resultant graphics.

Program Operation

To begin the program operation, the *reentry.m* file must be opened in the MATLAB® current directory. The program will create some **.mat* files in order to save the data and will delete them after the operation ends. Typing "reentry" will initiate the program and display the graphical user interface (GUI) menu. Figure 28 is the reentry GUI that will come up after starting the program operation. On the left hand side, the latitude and longitude are the desired landing coordinates that are expressed in WGS84 coordinate system. The coordinates are in degrees and can be selected between either 0E to 360E or 180W to 180E. However, west coordinates must be writes as negative numbers. Under the coordinates, the atmospheric entry time is displayed. As mentioned in previous chapter, the atmospheric entry time is based on the conceptual lunar departure time and it is created under the assumption of the beginning of 24 hour period is when the landing coordinate lines up with the lunar departure location. The Earth's tilt angles

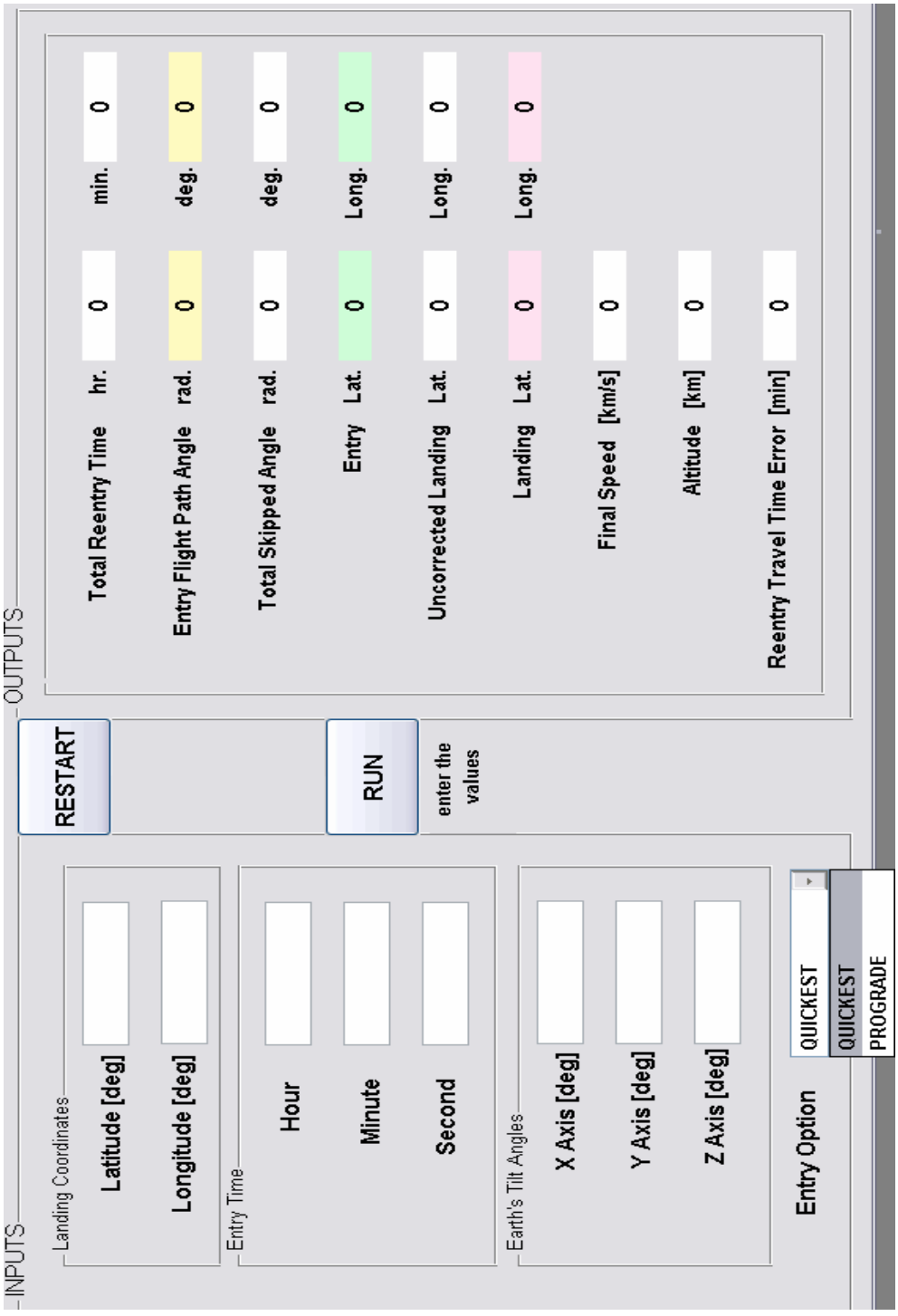

Figure 28. GUI display for reentry program

are defined according to the position of the Moon against Earth's ECEF coordinates where the tilt angles are the calculated counter clockwise in ECEF coordinates which could make the Earth's equator aligned with the orbit of the Moon. After entering all of the data, the reentry option selection can be made. The default option is adjusted to be the quickest entry type, however, the entry type can also be changed to prograde entry. The calculations will be made according to the selection. Pushing on the "RUN" button will start the process.

The result of the calculations will be displayed on the right hand side of the GUI display. The solution parameters are entry latitude, longitude, and the flight path angle at the entry altitude of 122 km. Other parameters displayed on the GUI are for information purposes. Final speed and altitude are the final parameters that are calculated by the program. The program ends its calculations when the CEV achieves the altitude 10 km. and gives the vehicle's final speed at that altitude. Uncorrected landing coordinates are to show the landing point with no coordinate rotation done when the tilt angles are ignored. Therefore, if the tilt angles are chosen to be zero, it will be the same as landing coordinates. At the end, if the "RESTART" button is pushed, the program will return to the beginning, closing all of the figures and deleting the inputs and outputs.

Software System Process

The MATLAB® codes developed in this research works by iterating the entry flight path angle to be able to find the right landing location. A detailed schema of the program can be seen in Figures 29 and 30.

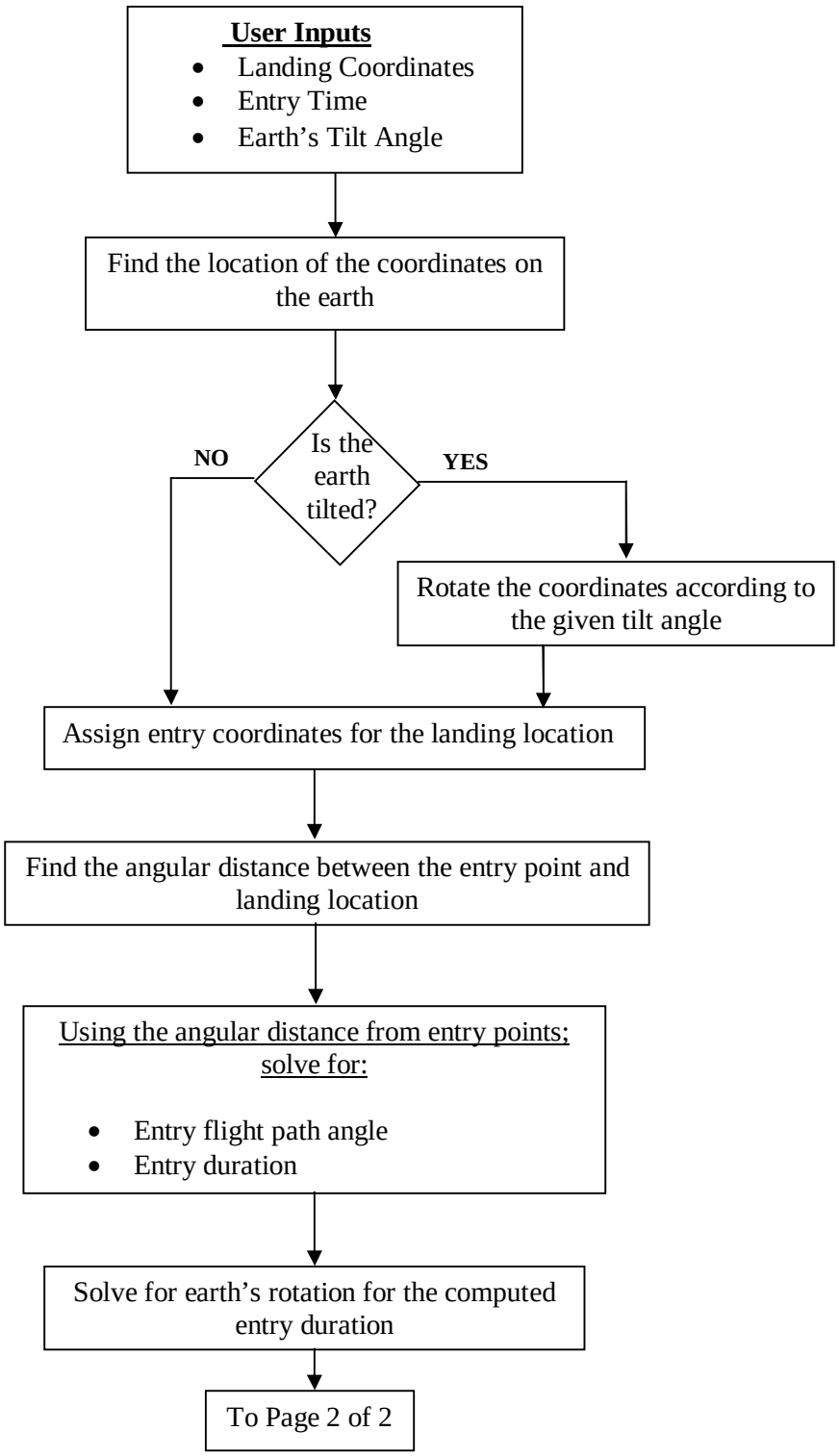

Figure 29. Software System Process Schema 1 of 2

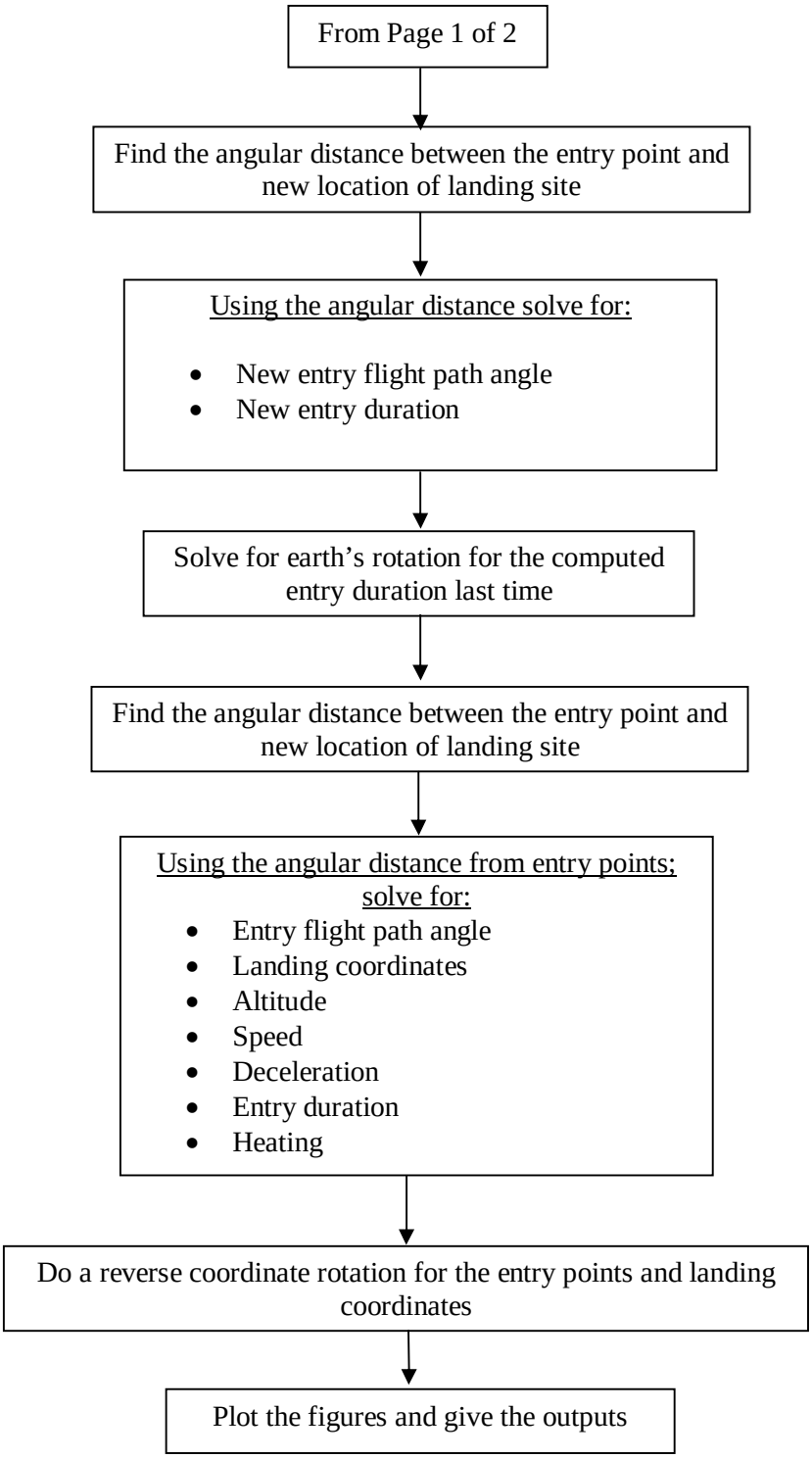

Figure 30. Software System Process Schema 2 of 2

After entering the desired landing coordinates, entry time and, tilt angles of the Earth, the program finds the exact location of the coordinates on the Earth. If the Earth is tilted, the coordinates will be rotated and the landing location will be expressed in the new coordinate system. Next, the entry coordinates are found to be used for the solution. After finding the entry coordinates, the flight distance could be found using the entry and landing location using spherical distance formulas as in Eqs.3.6, 3.7, and 3.8. Then, the program finds the entry flight path angle for the flight distance. Since it is very hard to reverse integrate the universal equations, the program uses a certain preassigned value for the beginning and starts iterating until the right flight path angle is found for the distance. Generally, the skip entry takes from 40 minutes up to 2 hours; therefore, the rotation of the Earth during the atmospheric entry should be calculated and added to the total rotation. Thus, the program adds the entry duration to the total time and finds a new location and a new flight distance. This iteration is done for three times to be able to reach the exact location of the landing coordinates and decrease the uncalculated rotation of the Earth during the entry flight. All of the parameters are calculated integrating the universal equations and the results are then converted into the usable parameters for the user. A reverse rotation of the coordinate system is done after the calculation is done for plotting the figures and giving the output coordinates. Finally, the results are displayed on the right hand side of the GUI display.

Results Analysis

In this section, the outputs of the program will be presented and a sample entry profile will be analyzed. The sample inputs and the outputs are seen in Figure 31.

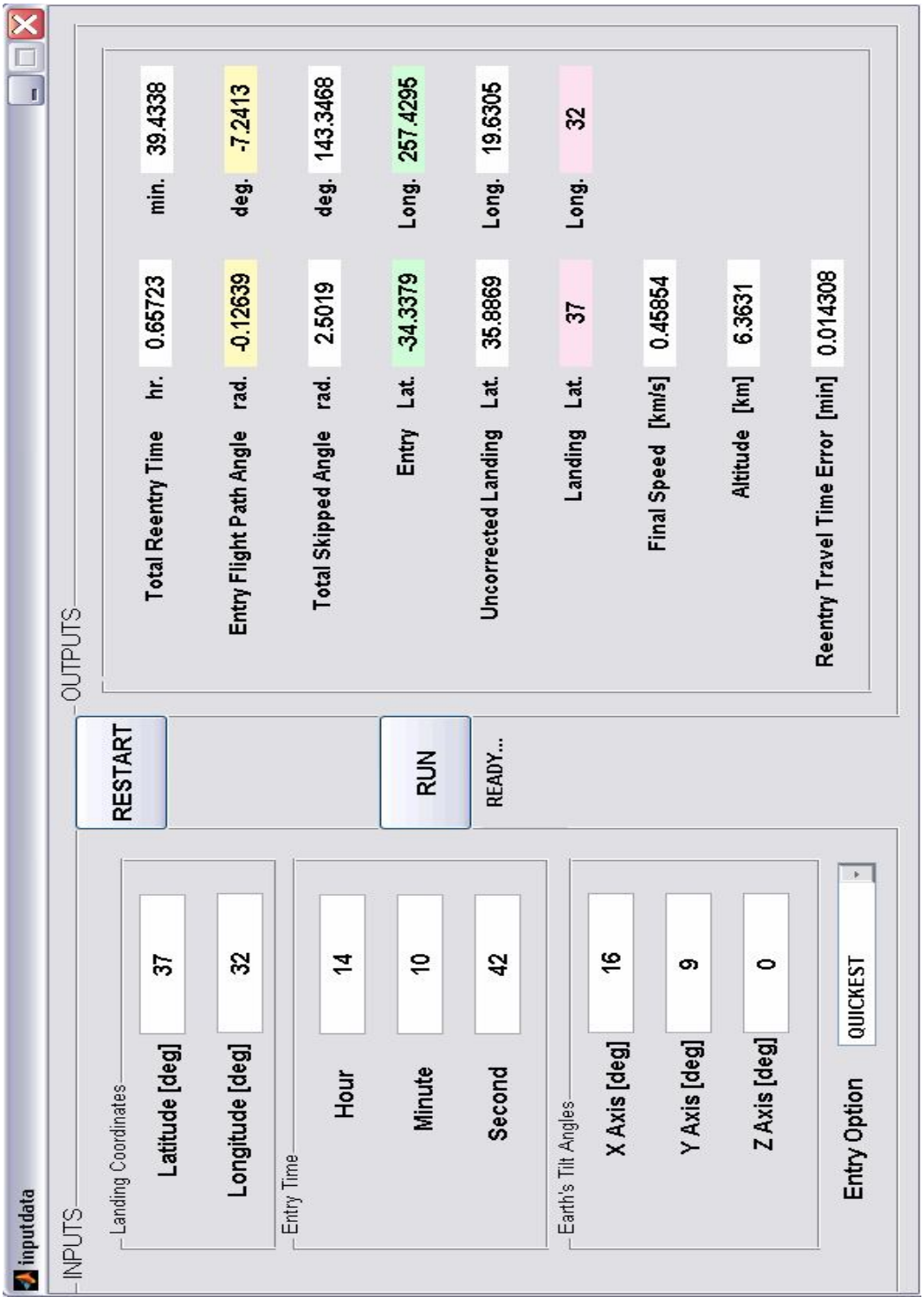

Figure 31. Total Skipped Longitude and Distance

The algorithm of the program is compatible with any coordinates on the Earth surface. So that, the calculations can be made regardless of the current ground facilities or landing sites available.

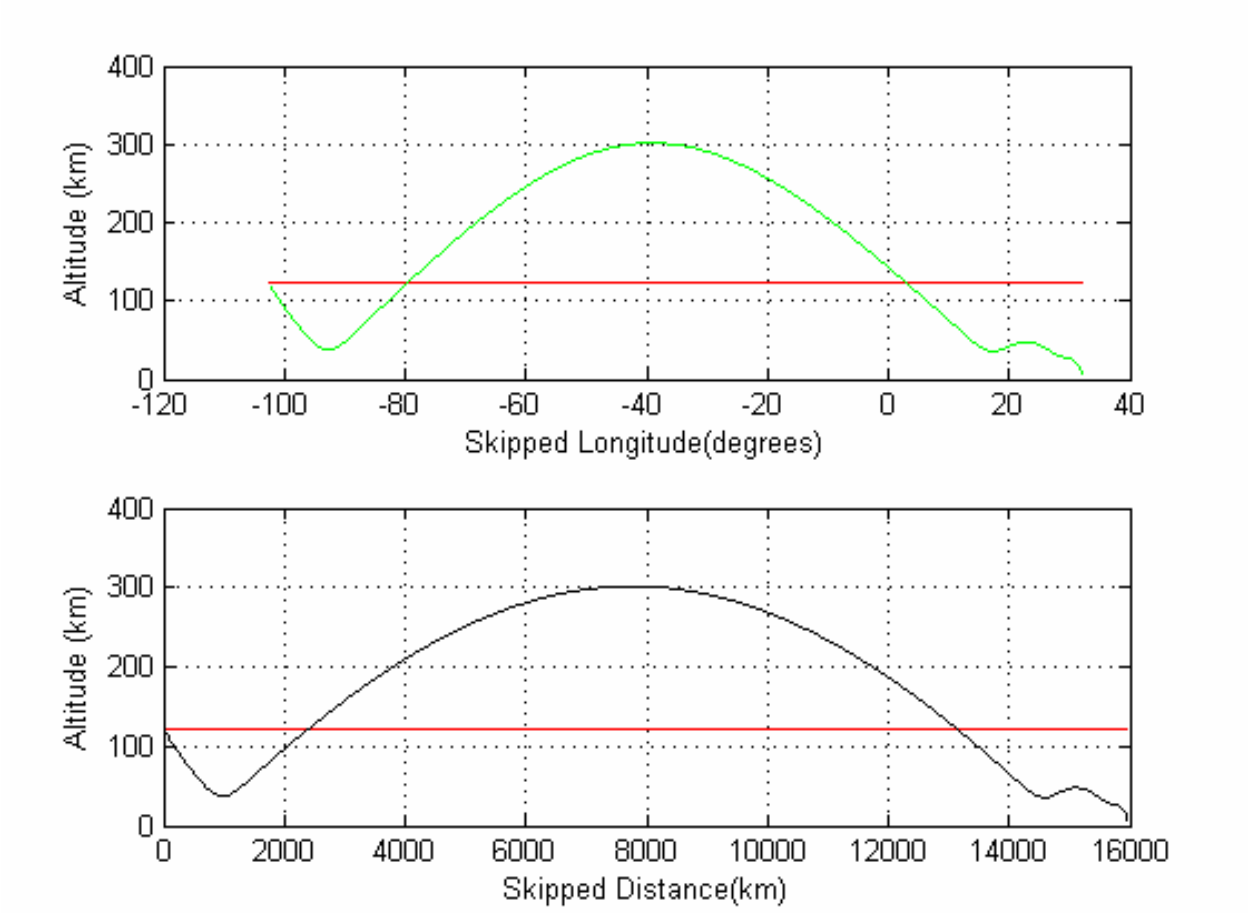

Figure 32. Total Skipped Longitude and Distance

Figure 32 shows the skipped distance and longitude vs. altitude. As seen in the graphs, the skipping altitude goes up to 300 km. and the flight distance reaches to a 16000 km. range. In both figures, the thin red line represents the atmosphere line, and it is also the entry altitude. The graphs do not match exactly since the distance between the

longitudes changes with latitude; therefore, the polar type entry graph looks rectangular. At the first entry, the CEV flies down to 45 km altitude before gaining a positive flight path angle. After the skip, the vehicle spends most of its trajectory out of the atmosphere. As far as the heating constraints, this is very helpful for cooling down the vehicle out of the atmosphere and beginning a second entry with less energy.

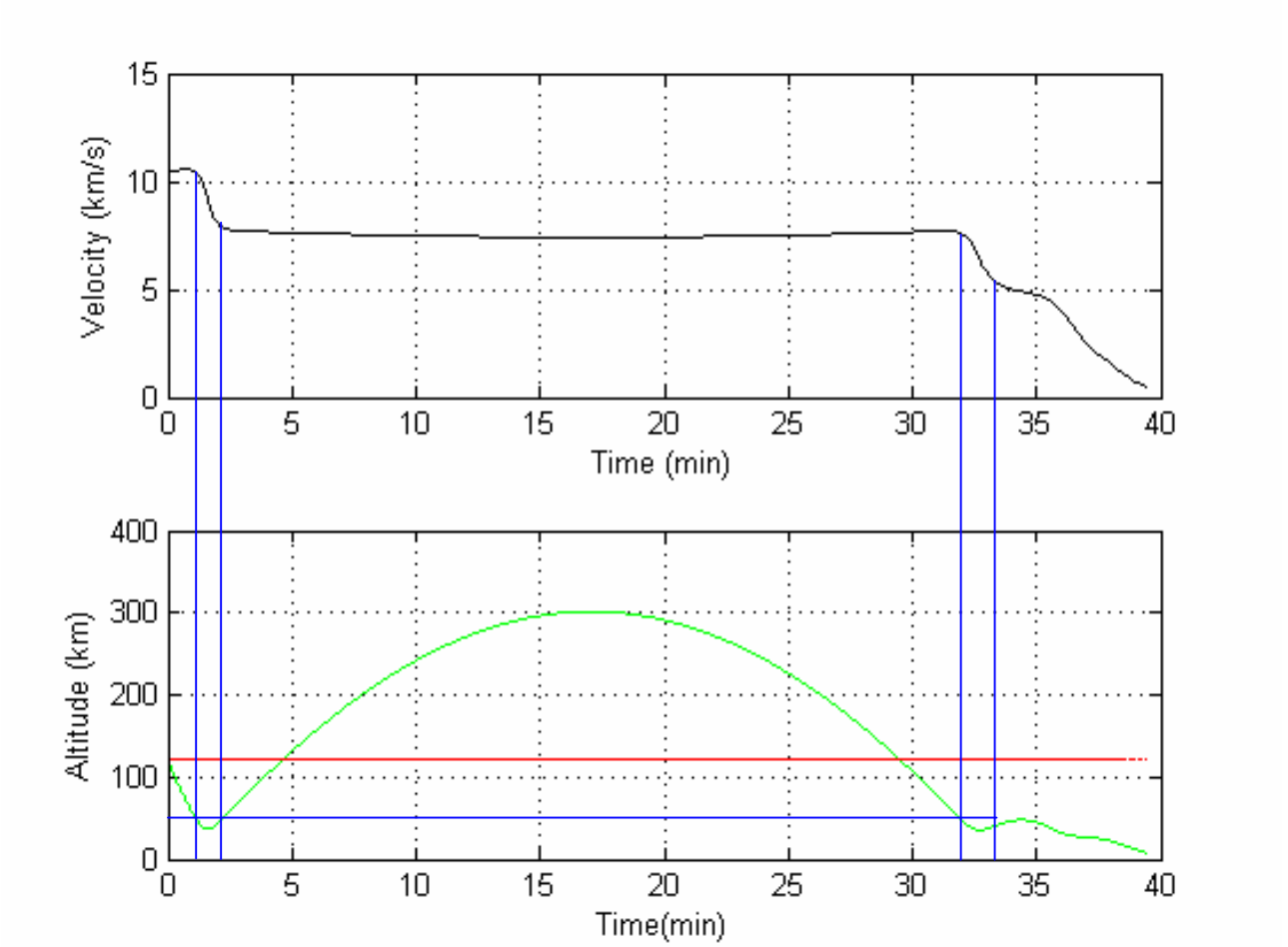

Figure 33. Velocity- Altitude Projection

In Figure 33, the change in velocity according to the time is projected on the altitude to be able to see how the speed changes in the atmosphere. As it is expected, the

speed of the vehicle tends to increase at the entry, and then it gradually starts decreasing. However, as seen on the line, the velocity of the vehicle is essentially the same as entry speed (10.5 km/s) even at 50 km. altitude. The deceleration on the vehicle increases because of the increased air density and drag afterwards and the vehicle loses most of its energy under that altitude during first and the second entry.

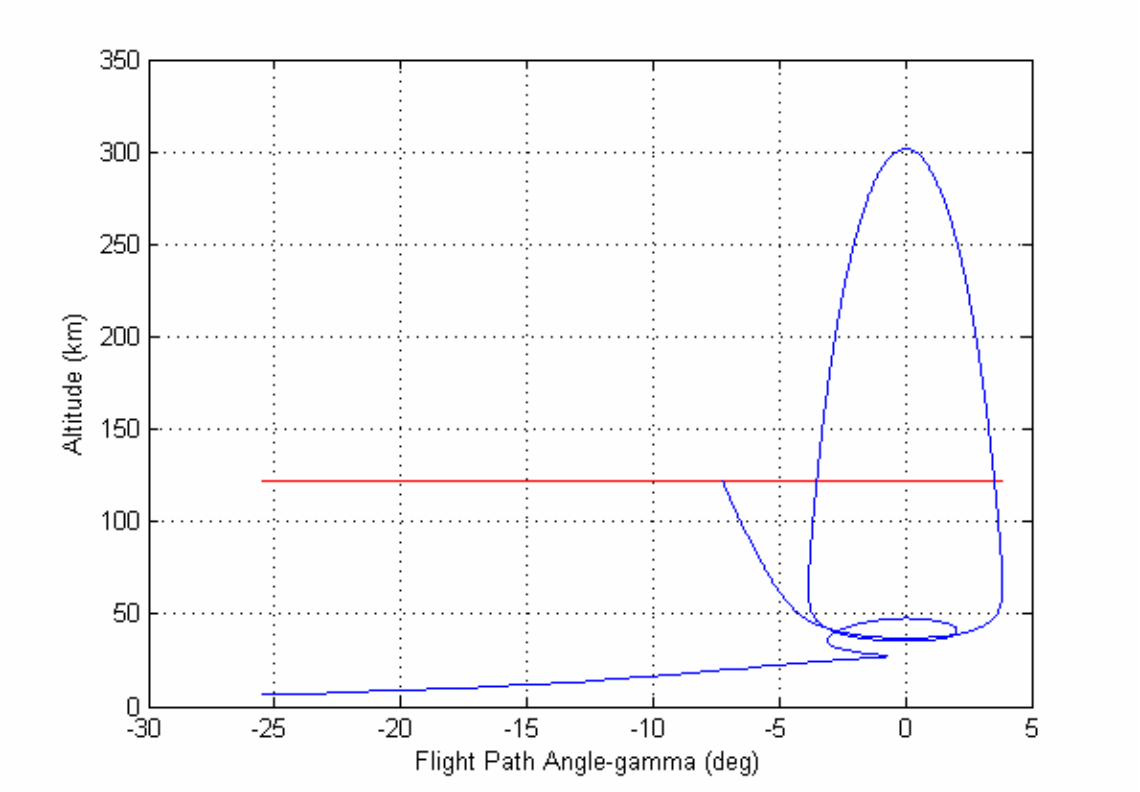

Figure 34. Flight Path Angle vs. Altitude

In Figure 34, the change in flight path angle shows the characteristics of the trajectory during the entry and the skip part. Since there is no perturbing force affecting the vehicle during the time between the base of the first skip and the base of the second entry point the flight path angle displays a symmetrical behavior.

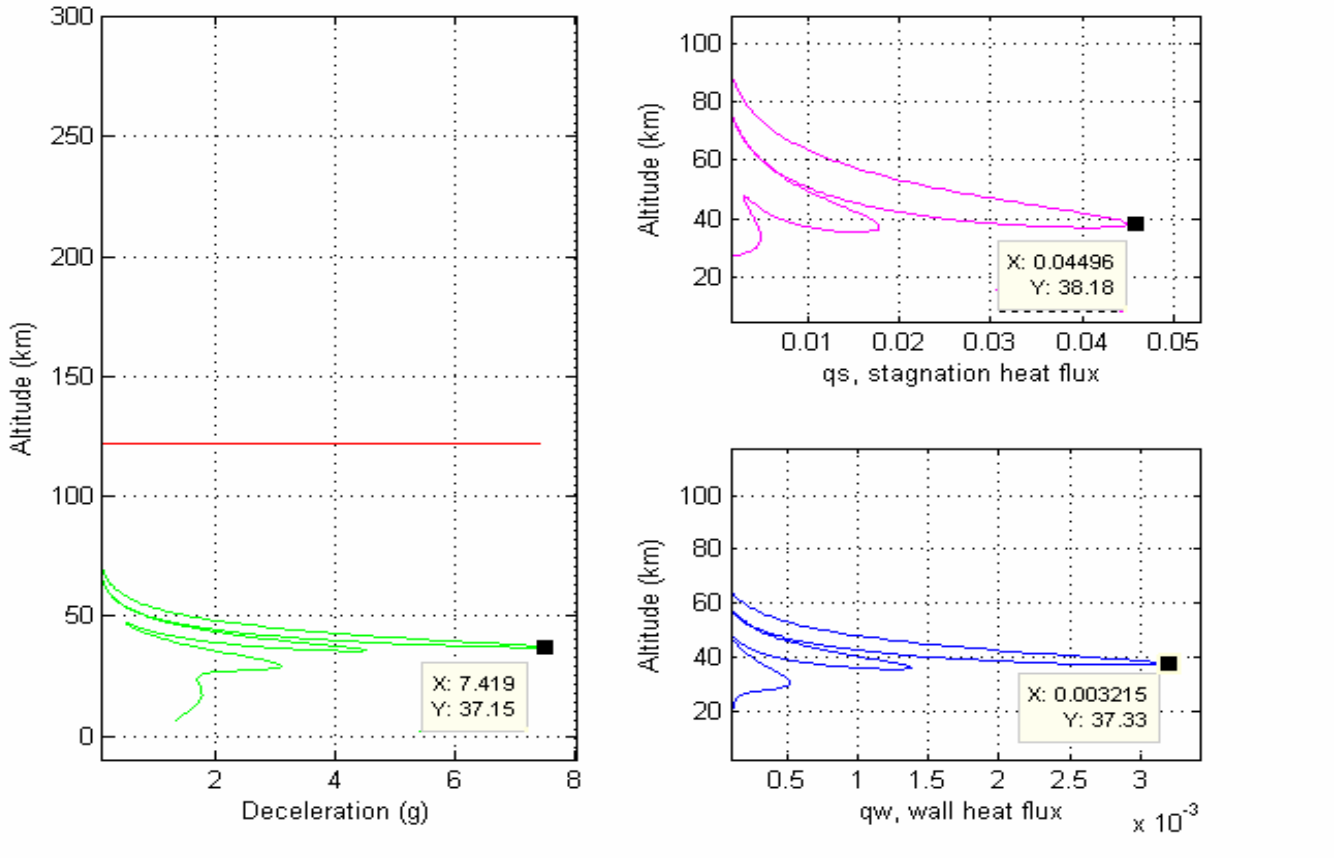

Figure 35. Deceleration, Stagnation, and Wall Heat Flux

In Figure 35, the deceleration, stagnation, and wall heat flux vs. altitude diagrams are presented. As it can be seen in the figures, the stagnation heat flux starts increasing in higher altitudes where the vehicle first meets the drag force but the wall heat flux increases with a higher rate and peaks right before maximum deceleration rate is achieved. The stagnation heat flux is the local "hot spot" on the vehicle where the wall heat flux is an average value on the vehicle. Therefore, it is expected that the stagnation heat flux peaks before wall heat flux.

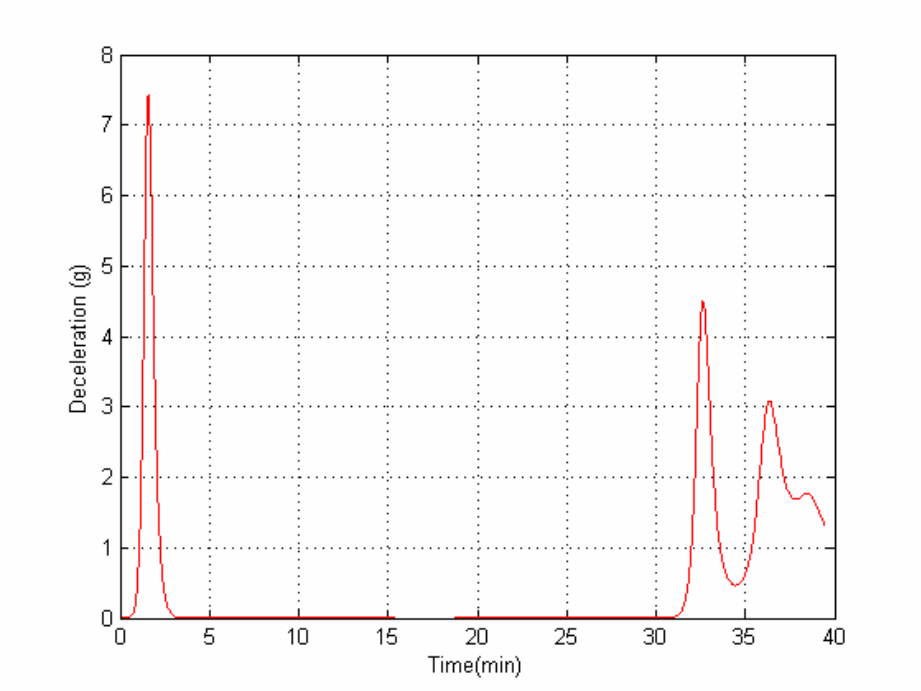

Figure 36. Deceleration vs. Time

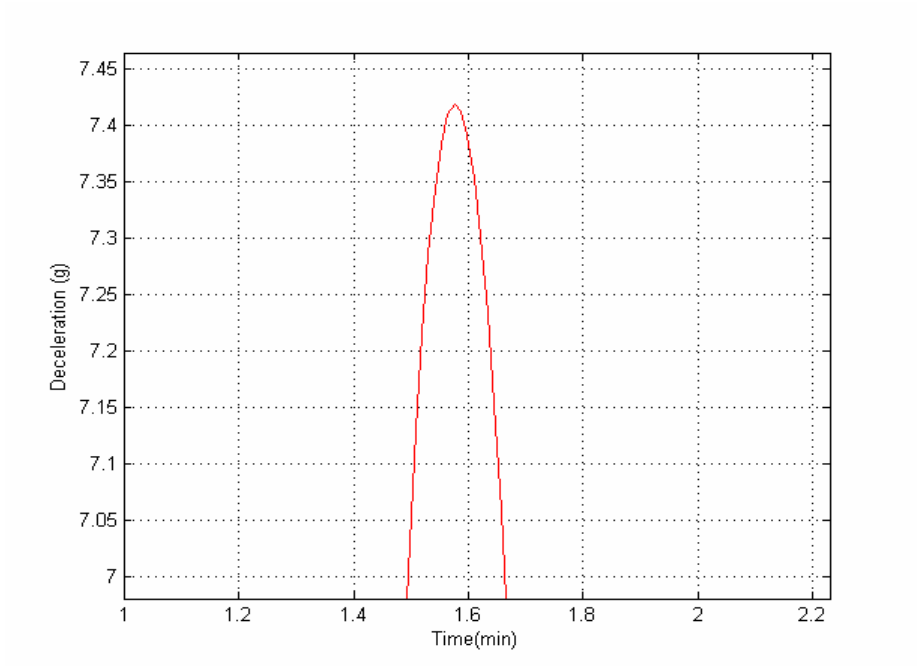

Figure 37. Maximum Deceleration vs. Time

In Figure 36 and 37 the deceleration and the maximum deceleration times can be seen. The maximum deceleration in this sample is 7.42 g., and the time above 7 g is 10 seconds. Depending on the crew seating positions, the maximum deceleration that a crew member can handle varies. However, 10 seconds over 7 g. and a maximum of 7.42 g. is lower than the NASA allowable deceleration limits which is 10 g. for up to 40 seconds. [16]

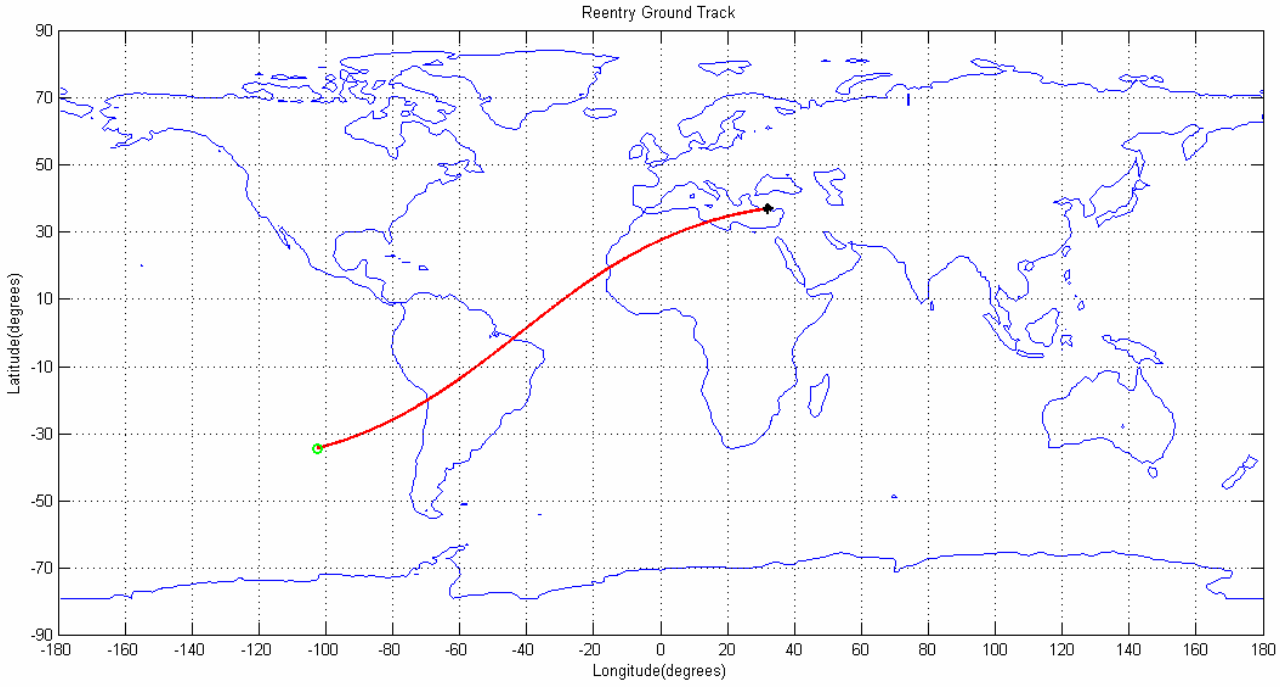

Figure 38. Ground Track of the Trajectory

In Figure 38, the ground track of the trajectory can be seen on a prograde reentry for the same example. The green circle represents the entry point, and black dot, the landing point. The program also calculates the possible entry coordinate errors according to the entry point.

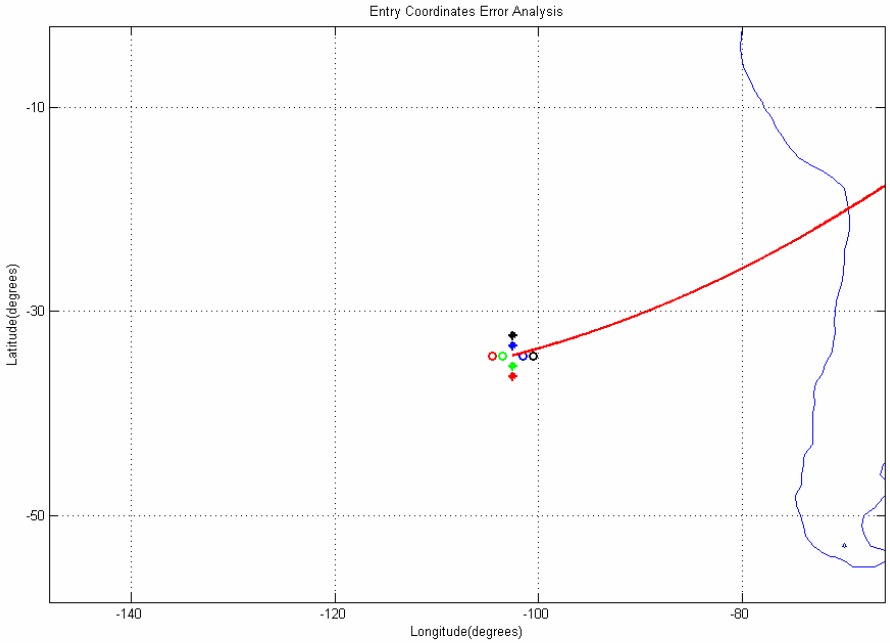

Figure 39. Reentry Coordinate Errors

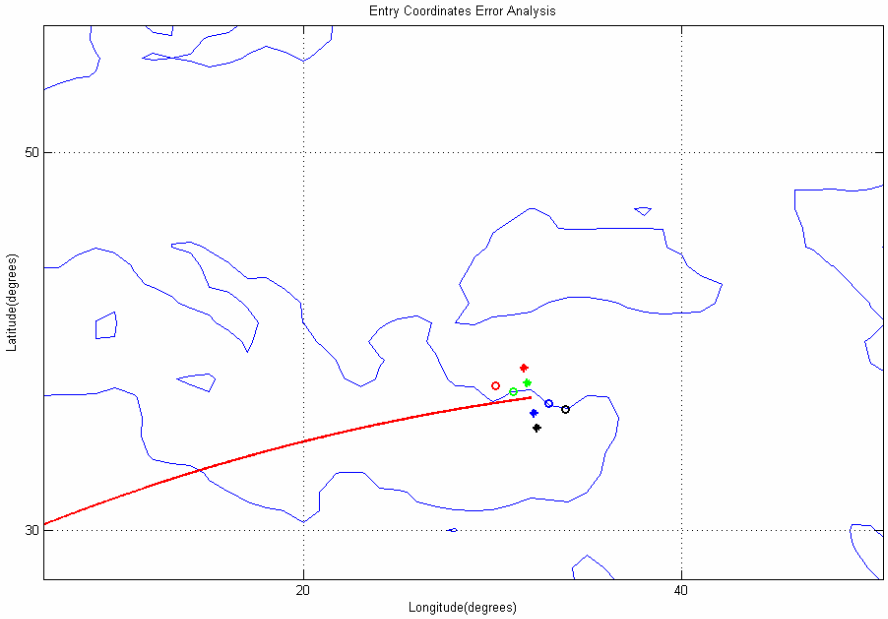

Figure 40. Landing Errors

Figure 39 and 40 presents the coordinate errors and the results. Eight different entry coordinate errors are given to the program to see how they affect the landing coordinates. The corresponding marks and colors are the results of the entry coordinate errors. Table 3 shows the entry and the landing coordinates solution.

Table 3. Entry / Landing Coordinate Errors (Lat/Long)

	Entry Coordinates (deg)	Landing Coordinates (deg)
Normal Entry	-34.3379 / 257.4295	37.0000/ 32.0000
One Long. West Entry	-34.3379 / 256.4295	37.3136 / 31.0627
Two Long. West Entry	-34.3379 / 255.4295	37.6274 / 30.1216
One Lat. South Entry	-35.3379 / 257.4295	37.7897 / 31.8218
Two Lat. South Entry	-36.3379 / 257.4295	38.5775 / 31.6292
One Long. East Entry	-34.3379 / 258.4295	36.6867 / 32.9336
Two Long. East Entry	-34.3379 / 259.4295	36.3738 / 33.8637
One Lat. North Entry	-33.3379 / 257.4295	36.2087 / 32.1644
Two Lat. North Entry	-32.3379 / 257.4295	35.4158 / 32.3153

Summary

In this chapter, the process executing the program and its results are presented. The results and the graphics can be changed and displayed as the needs for the outputs change. Although the program does not put any restrictions on the process to keep the

deceleration loads under certain values, the design of the algorithm naturally avoids high deceleration rates occurring during the reentry. For heat and deceleration concerns, the highest values are reached during the short distance trajectories, since the vehicle has to dissipate more of its energy in less time. Thus, the algorithm selects a trajectory, which has a minimum of 45 degrees skipping distance to avoid high deceleration and heating values.

The program operation takes a few minutes because of the iteration of the ODE function in MATLAB® environment but it is completely dependent on the selected coordinates. If the flight distance is close to 360 degrees, the limits on the integration used in ODE function goes higher linearly and the solution takes more time. However, this is also strictly dependent on the processor speed of the computer.

The program is designed to be as user friendly as possible, therefore the input parameters and the results are displayed in the same window. Since the program is composed of many small functions, it is easy to change any part depending on the needs and future developments.

V. Conclusions and Recommendations

Conclusions of Research

In this research, the effects, and benefits of a skip entry trajectory is inspected and for this reason, a MATLAB® program is developed. The main reason for the program is to be able to show that a skip entry trajectory from a lunar return mission in a manned space capsule is possible and has many benefits comparing to Apollo type trajectories. First and the most important benefit of this trajectory is its independence to the lunar departure time constraints. This can be a result of an emergency during the mission or an early or late completion. Therefore, in order to achieve a safe landing from the mission return, the skip entry trajectory provides a safe and time independent solution. Another benefit is also its independence from the landing site. However, this is not a complete independence. Landing coordinates have to be decided as early as possible since the landing site selection makes the trajectory dependent upon entry coordinates and flight path angle. These selections have to be made early in order to save fuel and reach the entry parameters. This situation can be considered as a con, but it still gives more freedom than having to leave at a specific time to be able to land at the right spot.

Another important reason for the program is to look at the trade-offs in the trajectory and the vehicle parameters. It is easy to change the vehicle parameters or the entry conditions to try different reentry solutions for the changing needs. New trajectories or vehicle types can be implemented depending on the mission characteristics.

Significance of Research

In general, it is considered that the results of the research are quite successful. Although total skip entry guidance has been done before for the trajectories of unmanned vehicles, application of this concept is quite new for the manned space missions. One very important risk of trying a skip entry is skipping out of the atmosphere above orbital escape speed at that altitude and this can have disastrous results. Therefore, the accuracy in maintaining the parameters is very important as calculating the correct parameters.

The skip entry guidance concept is going to be a part of the CEV reentry algorithm. Since the vehicle and its guidance system is still under development and no public displays or announcements have been made so far, any kind of different perspective and approach to the problem will be helpful in terms of putting more insight for the solution of the problem.

Recommendations for Future Research

The MATLAB® program developed for this problem is quite adaptable for future developments. Several areas can be improved in this research. Some of those are neglected for the simplification purposes but some of them led the problem in different areas of expertise. Therefore, the general solution method should be developed using interdisciplinary research methods.

In this research, the atmosphere is modeled with a simple exponentially increasing atmosphere type. Although it can be neglected and does not change the results significantly, the accuracy of the program can be improved by modeling the atmosphere layer by layer, each with a different scale height.

Atmospheric events are also neglected during this research for simplification purposes. Although the events cannot be estimated beforehand, a major factor as an average can be used in future developments. The rotation of the Earth is added to the calculation during the time spent in the atmospheric trajectory, but the atmosphere is assumed to be inertially fixed. However, it is also known that the lower layers of the atmosphere are rotating with the Earth and exponentially decreasing as the altitude increases. This concept is not hard to model and can be implemented in the calculations.

The lift and drag coefficients are assumed to stay constant during the reentry. However, as the temperature and the aerodynamic pressure rise on the body of the vehicle, its aerodynamics tend to change the lift and drag coefficients of the vehicle slightly. This is also neglected because of its insignificant effects on the total result. However, it can also be modeled and included to the calculations in terms of increasing the accuracy.

The angle of attack and the bank angle used in this solution are held constant throughout the trajectory. Under perfect conditions, it does not cause any problems, however, the equations used in this research are developed for spherical entry and other equations, which include the obliqueness of the Earth, had to be simplified to the spherical versions. Therefore, if the same set of reentry equations are used, the drift caused by the obliqueness of the Earth has to be compensated by changing either the bank angle or the flight path angle, or both, considering all of the other conditions is perfect.

Appendix

reentry.m

```
% ASSUMPTIONS
% landing coordinates are assumed to be alligned with the lunar departure
%at time 00:00:00, entry time will be defined according to this assumption
% the earth is perfectly spherical
% atmospheric density is decreasing exponentially
% entry speed (V_e) is assumed to be 10.5 km/sec
% the vehicle mass is assumed to be 11500 kg.
% other constants regarding to the CEV are taken from NASA project documents
% earth's tilt angle is measured counter-clockwise direction on each axis
%in ECEF coordinates

close all;
clear all;
clc;

% initial conditions fot the gui display

entrytimehr=0;
save('entrytimehr');
entrytimemin=0;
save('entrytimemin');
fparad=0;
save('fparad');
fpadeg=0;
save('fpadeg');
entrylat=0;
save('entrylat');
entrylong=0;
save('entrylong');
skippedrad=0;
save('skippedrad');
skippeddeg=0;
save('skippeddeg');
uncorrlat=0;
save('uncorrlat');
uncorrlong=0;
save('uncorrlong');
landlat=0;
save('landlat');
```

```
landlong=0;
save('landlong');
finspeed=0;
save('finspeed');
alt=0;
save('alt');
timerr=0;
save('timerr');
please='enter the values';
save('please');
pro='QUICKEST';
save('pro');

% running gui screen and data input
inputdata;
uiwait(inputdata);      % inputs wait until run button is pressed

% taking input values from saved files
load('lat');
load('long');
load('ho');
load('min');
load('sec');
load('x');
load('y');
load('z');
load('pro');

phi_l_deg= lat;
theta_l_deg= long;
hour= ho;
minute= min;
second= sec;
x_deg= x;
y_deg= y;
z_deg= z;

% Function tilthange changes the landing coordinates according to the tilt
% angle between earth's equator and lunar orbit

[theta_l,phi_l] = tiltchange(theta_l_deg, phi_l_deg, x_deg, y_deg, z_deg);


day=86400;                                  % 1 day in seconds
time=hour*3600+minute*60+second;            % current time in seconds
```

```
rot=time*(2*pi)/day;                    % earth's rotation during given time
center_theta=theta_l-rot;               % center longitude


if rot>(pi/2) & rot<(3*pi/2);           % if the rotation is greater than 90 degrees
   center_theta=center_theta+pi;        % it is changing the center longitude to the other
end;                                    % side


if rot>=(3*pi/2);
  center_theta=center_theta+2*pi;
end;

C=pi/2;                                 % angle between equator & landing longitude
a=phi_l;                                % landing latitude angle
b=abs(theta_l-center_theta);            % difference between center and landing
                                        % longitude
c=acos(cos(a)*cos(b));%+sin(a)*sin(b)*cos(C))  % angular distance between center and
                                               % landing point
A=asin((sin(a)*sin(C))/sin(c));         % inclination (angle between the orbit and
                                        % equator)
inc=A;                                  % inclination (angle between the orbit and
                                        % equator)


% This function is for determination of the entry parameters

[theta_e,phi_e,s_end] = entryoption(rot,center_theta,phi_l,theta_l,inc,pro);

if hour<24;          % The program is defined in one day timezone

  %initial conditions for the program
  color=['g','k','m','b','c'];

  %define constants:
  m=11500;             % mass
  Beta=0.14;           % scaling height
  rho_s=1.225e9;       % atmospheric density at the surface
  S=23.76e-6;          % entry surface area of the CEV
  Cd =.11;             % drag coefficient of CEV
  mu=398600;           % erath's gravitational parameter
  ltd=0.4 ;            % lift to drag ratio
  gamma_e= -0.1095;    % reentry flight path angle
  s_end
```

```
% Defining starting conditions for the flight path angle
if s_end>1.576 & s_end<3.152;
   gamma_e=-0.125
elseif s_end>=3.152 & s_end<5.507;
   gamma_e=-0.120
elseif s_end>=5.507 & s_end<6.521;
   gamma_e=-0.114
elseif s_end>=6.521 & s_end<=7;
   gamma_e=-0.110
end;

% All the computations are made until the vehicle flies below 10 km.
altitude=30;
while altitude(end,end)>10;

% Finding an increase rate for the iteration of gamma_e in order to  get it fast

  if s_end>1.576 & s_end<5.507;
     if altitude(end,end)<35;
       gamma_e=gamma_e-0.000005
       else gamma_e=gamma_e-0.0001
     end;
  end;
  if s_end<=1.576;
     if altitude(end,end)<30;
     gamma_e=gamma_e-0.000005
     else gamma_e=gamma_e-0.0001
     end;
  end;
  if s_end>=5.507  & s_end<6.521;
     if altitude(end,end)<30;
     gamma_e=gamma_e-0.00001
     else gamma_e=gamma_e-0.0001
     end;
  end;
  if s_end>=6.521 & s_end<8;
     if altitude(end,end)<30;
     gamma_e=gamma_e-0.000005
     else gamma_e=gamma_e-0.00005
     end;
  end;
  if s_end >=8 & s_end<9.07;
     if altitude(end,end)<40;
     gamma_e=gamma_e-0.000005
     else gamma_e=gamma_e-0.00002
```

```
   end;
end;
if s_end >=9.07;
   if altitude(end,end)<35;
   gamma_e=gamma_e-0.000003
   else gamma_e=gamma_e-0.00001
   end;
end;

%define initial conditions
Ve=10.5;                                % reentry speed
re=6500;                                % reentry radius
he=re-6378;                             % reentry altitude
Ze=rho_s*exp(-Beta*he)*Cd*S/2/m*sqrt((he+6378)/Beta); % (eq 9.22)
ue=Ve^2*(cos(gamma_e))^2*re/mu;                  % (eq 9.21)
psi_e=0;                                % initial heading angle

s_i=0;                                  % initial s value for the integration
s_f=s_end;                              % final s value for the integration
t0=0;                                   % initial t value for the time solution
x0=[Ze ue theta_e phi_e gamma_e psi_e t0];          % entry conditions for the
                                                    %  integration
Br=900;                                 % BetaR value is assumed to be constant with
                                        % exponantially changing atmosphric properties
sigma= 0;                               % bank angle is zero durin the trajectory
lift_to_drag=ltd;                       % CEV constant lift to drag ratio

options = odeset('MaxStep',0.001);              % setting step size of the ODE
                                                %  function

% solving differential equations with solver function
% (eq 9.29/9.30/9.31/9.32/9.33/9.34)
[s,x]=ode23(@solver,[s_i s_f],x0,options,Br,sigma,lift_to_drag,…
        mu,m,Beta,rho_s,S,Cd);

% arrangement of the outputs
Z=x(:,1);
u=x(:,2);
theta=x(:,3);
phi=x(:,4);
gamma=x(:,5);
psi=x(:,6);
t_time=x(:,7);

eta=Z/sqrt(Br);          %with assumption of BetaR constant
```

```
    altitude=1/-Beta*log(eta*2*m*Beta/rho_s/S/Cd);

  end;

 timecorrection1(phi_l_deg,theta_l_deg,time,x_deg,y_deg,z_deg,t_time,hour,minute,…
 second,gamma_e,pro);

else
   % if the input entry time is out of the 24 hour day period
   clc;
   disp(' ');
   fprintf('Entry time: %2.0f:%2.0f:%2.0f is NOT acceptable!\n',hour,minute,second);
   disp(' ');
   fprintf('PLEASE RE-RUN THE PROGRAM AND ENTER A CORRECT TIME!\n ');
   disp(' ');
end
```

function timecorrection1.m

```
function timecorrection1(phi_l_deg,theta_l_deg,time,x_deg,y_deg,z_deg,t_time…
,hour,minute,second,gamma_e,pro)
clc;

% Function tilthange changes the landing coordinates according to the tilt
% angle between earth's equator and lunar orbit
[theta_l,phi_l] = tiltchange(theta_l_deg, phi_l_deg, x_deg, y_deg, z_deg);

timeold=t_time(end,end);
day=86400;                          % 1 day in seconds
time=time+t_time(end,end);          % current time in seconds
rot=time*(2*pi)/day;                % earth's rotation during given time
center_theta=theta_l-rot;           % center longitude


if rot>(pi/2) & rot<(3*pi/2);       % if the rotation is greater than 90 degrees
   center_theta=center_theta+pi;    % it is changing the center longitude to the other
end;                                % side

if rot>=(3*pi/2);
  center_theta=center_theta+2*pi;
end;

C=pi/2;                             % angle between equator & landing longitude
a=phi_l;                            % landing latitude angle
```

```
b=abs(theta_l-center_theta);                % difference between center and landing
longitude
c=acos(cos(a)*cos(b));%+sin(a)*sin(b)*cos(C))  % angular distance between center and
                                                % landing point
A=asin((sin(a)*sin(C))/sin(c));             % inclination (angle between the orbit and
                                            % equator)
inc=A;                                      % inclination (angle between the orbit and
                                            % equator)

% This function is for determination of the entry parameters
[theta_e,phi_e,s_end] = entryoption(rot,center_theta,phi_l,theta_l,inc,pro);

  %initial conditions for the program
  altitude=30;
  color=['g','k','m','b','c'];

  % define constants:
  m=11500;              % mass
  Beta=0.14;            % scaling height
  rho_s=1.225e9;        % atmospheric density at the surface
  S=23.76e-6;           % entry surface area of the CEV
  Cd =.11;              % drag coefficient of CEV
  mu=398600;            % erath's gravitational parameter
  ltd=0.4 ;             % lift to drag ratio

  % Defining starting conditions for the flight path angle
  if pro=='QUICKEST'
    if time>43200
      gamma_e=gamma_e+0.001
    else
      gamma_e=gamma_e+0.0005
    end;
  else
    if s_end>1.576 & s_end<3.152;
      gamma_e=-0.125
    elseif s_end>=3.152 & s_end<5.507;
      gamma_e=-0.120
    elseif s_end>=5.507 & s_end<6.521;
      gamma_e=-0.114
    elseif s_end>=6.521 & s_end<=7;
      gamma_e=-0.110
    end;
  end
```

```
% All the computations are made until the vehicle flies below 10 km.
while min(altitude)>10;

   % Finding an increase rate for the iteration of gamma_e in order to
   % get it fast
    if s_end>1.576 & s_end<5.507;
       if min(altitude)<35;
       gamma_e=gamma_e-0.000005
       else gamma_e=gamma_e-0.0001
       end;
   end;
   if s_end<=1.576;
       if min(altitude)<30;
       gamma_e=gamma_e-0.000005
       else gamma_e=gamma_e-0.0001
       end;
   end;
   if s_end>=5.507  & s_end<6.521;
       if min(altitude)<30;
       gamma_e=gamma_e-0.00001
       else gamma_e=gamma_e-0.0001
       end;
   end;
   if s_end>=6.521 & s_end<8;
       if min(altitude)<30;
       gamma_e=gamma_e-0.000005
       else gamma_e=gamma_e-0.00005
       end;
   end;
   if s_end >=8 & s_end<9.07;
       if min(altitude)<40;
       gamma_e=gamma_e-0.000005
       else gamma_e=gamma_e-0.00002
       end;
   end;
   if s_end >=9.07;
       if min(altitude)<35;
       gamma_e=gamma_e-0.000003
       else gamma_e=gamma_e-0.00001
       end;
   end;

   %define initial conditions
   Ve=10.5;                              % reentry speed
```

```
    re=6500;                                % reentry radius
    he=re-6378;                             % reentry altitude
    Ze=rho_s*exp(-Beta*he)*Cd*S/2/m*sqrt((he+6378)/Beta);  % (eq 9.22)
    ue=Ve^2*(cos(gamma_e))^2*re/mu;                        % (eq 9.21)
    psi_e=0;                                               % initial heading angle

    s_i=0;                                % initial s value for the integration
    s_f=s_end;                            % final s value for the integration
    t0=0;                                 % initial t value for the time solution
    x0=[Ze ue theta_e phi_e gamma_e psi_e t0];          % entry conditions for the
                                                        % integration
    Br=900;                               % BetaR value is assumed to be constant with
                                          % exponentially changing atmospheric
                                          % properties
    sigma= 0;                             % bank angle is zero during the trajectory
    lift_to_drag=ltd;                     % CEV constant lift to drag ratio

    options = odeset('MaxStep',0.001);      % setting step size of the ODE function

    % solving differential equations with solver function
    % (eq 9.29/9.30/9.31/9.32/9.33/9.34)
    [s,x]=ode23(@solver,[s_i s_f],x0,options,Br,sigma,lift_to_drag,…
          mu,m,Beta,rho_s,S,Cd);

    % arrangement of the outputs
    Z=x(:,1);
    u=x(:,2);
    theta=x(:,3);
    phi=x(:,4);
    gamma=x(:,5);
    psi=x(:,6);
    t_time=x(:,7);

    eta=Z/sqrt(Br);          %with assumption of BetaR constant
    altitude=1/-Beta*log(eta*2*m*Beta/rho_s/S/Cd);

  end;
  timenew=t_time(end,end);


timecorrection2(phi_l_deg,theta_l_deg,time,x_deg,y_deg,z_deg,timenew,timeold,hour,…
   minute,second,gamma_e,pro);
```

function timecorrection2.m

```
function timecorrection2(phi_l_deg,theta_l_deg,time,x_deg,y_deg,z_deg,timenew,…
timeold,hour,minute,second,gamma_e,pro)
clc;

% Function tilt change changes the landing coordinates according to the tilt
% angle between earth's equator and lunar orbit
[theta_l,phi_l] = tiltchange(theta_l_deg, phi_l_deg, x_deg, y_deg, z_deg);


day=86400;                          % 1 day in seconds
time=time+abs(timenew-timeold);     % current time in seconds
rot=time*(2*pi)/day;                % earth's rotation during given time
center_theta=theta_l-rot;           % center longitude


if rot>(pi/2) & rot<(3*pi/2);       % if the rotation is greater than 90 degrees
    center_theta=center_theta+pi;   % it is changing the center longitude to the other
 end;                               % side


if rot>=(3*pi/2);
   center_theta=center_theta+2*pi;
end;

C=pi/2;                             % angle between equator & landing longitude
a=phi_l;                            % landing latitude angle
b=abs(theta_l-center_theta);        % difference between center and landing
                                    % longitude
c=acos(cos(a)*cos(b));%+sin(a)*sin(b)*cos(C))  % angular distance between center and
                                               %  landing point
A=asin((sin(a)*sin(C))/sin(c));     % inclination (angle between the orbit and
                                    % equator)
inc=A;                              % inclination (angle between the orbit and
                                    % equator)

% This function is for determination of the entry parameters
[theta_e,phi_e,s_end] = entryoption(rot,center_theta,phi_l,theta_l,inc,pro);

  %initial conditions for the program
  altitude=30;
  color=['g','k','m','b','c'];
```

```
%define constants:
m=11500;              % mass
Beta=0.14;            % scaling height
rho_s=1.225e9;        % atmospheric density at the surface
S=23.76e-6;           % entry surface area of the CEV
Cd =.11;              % drag coefficient of CEV
mu=398600;            % erath's gravitational parameter
ltd=0.4 ;             % lift to drag ratio

% Defining starting conditions for the flight path angle
if time>21600 & time<64800
  gamma_e=gamma_e+0.001
else
  gamma_e=gamma_e+0.0005
end;


% All the computations are made until the vehicle flies below 10 km.
while min(altitude)>10;

   % Finding an increase rate for the iteration of gamma_e in order to
   % get it fast
   if s_end>1.576 & s_end<5.507;
      if min(altitude)<35;
      gamma_e=gamma_e-0.000005
      else gamma_e=gamma_e-0.0001
      end;
  end;
  if s_end<=1.576;
      if min(altitude)<30;
      gamma_e=gamma_e-0.000005
      else gamma_e=gamma_e-0.0001
      end;
  end;
  if s_end>=5.507  & s_end<6.521;
      if min(altitude)<30;
      gamma_e=gamma_e-0.00001
      else gamma_e=gamma_e-0.0001
      end;
  end;
  if s_end>=6.521 & s_end<8;
      if min(altitude)<30;
      gamma_e=gamma_e-0.000005
      else gamma_e=gamma_e-0.00005
      end;
```

```
    end;
    if s_end >=8 & s_end<9.07;
       if min(altitude)<40;
       gamma_e=gamma_e-0.000005
       else gamma_e=gamma_e-0.00002
       end;
    end;
    if s_end >=9.07;
       if min(altitude)<35;
       gamma_e=gamma_e-0.000003
       else gamma_e=gamma_e-0.00001
       end;
    end;

    %define initial conditions
    Ve=10.5;                                % reentry speed
    re=6500;                                % reentry radius
    he=re-6378;                             % reentry altitude
    Ze=rho_s*exp(-Beta*he)*Cd*S/2/m*sqrt((he+6378)/Beta);    % (eq 9.22)
    ue=Ve^2*(cos(gamma_e))^2*re/mu;                          % (eq 9.21)
    psi_e=0;                                % initial heading angle

    s_i=0;                                  % initial s value for the integration
    s_f=s_end;                              % final s value for the integration
    t0=0;                                   % initial t value for the time solution
    x0=[Ze ue theta_e phi_e gamma_e psi_e t0];          % entry conditions for the
                                                        % integration
    Br=900;                                 % BetaR value is assumed to be constant with
                                            % exponantially changing atmosphric
                                            % properties
    sigma= 0;                               % bank angle is zero durin the trajectory
    lift_to_drag=ltd;                       % CEV constant lift to drag ratio

    options = odeset('MaxStep',0.001);              % setting step size of the ODE
                                                    % function

    % solving differential equations with solver function
    % (eq 9.29/9.30/9.31/9.32/9.33/9.34)
    [s,x]=ode23(@solver,[s_i
s_f],x0,options,Br,sigma,lift_to_drag,mu,m,Beta,rho_s,S,Cd);

    % arrangement of the outputs
    Z=x(:,1);
    u=x(:,2);
    theta=x(:,3);
```

```
    phi=x(:,4);
    gamma=x(:,5);
    psi=x(:,6);
    t_time=x(:,7);

    theta_deg=rad2deg(theta);
    phi_deg=rad2deg(phi);
    psi_deg=rad2deg(psi);

    eta=Z/sqrt(Br);          %with assumption of BetaR constant
    altitude=1/-Beta*log(eta*2*m*Beta/rho_s/S/Cd);

  end;

  %Finding kinetic energy (T) (eq 9.89)
  T=(u/2)./((cos(gamma)).^2);

  %Finding deceleration (dec:g) (eq 6.76)
  dec=(2*Br*eta).*(T*sqrt(1+ltd^2));

  %Finding stagnation heat flux (eq 7.23)
  qs=(eta.^0.5).*(T.^1.5);

  %Finding wall heat flux (eq 7.20)
  qw=(eta).*(T.^1.5);

  % adjusting the graphic index numbers according to the tilt change with
  % change_graph function
  [theta_deg_nlg,phi_deg_nlg] = change_graph(phi_l,theta, phi, theta_e, x_deg, y_deg,
z_deg, time,pro);

  % Finding total atmospheric travel time and final velocity
  V=sqrt(u*mu./(altitude+6378))./cos(gamma);
  travelminute=t_time/60;
  totalminute=travelminute(end,end);
  totalhour=totalminute/60;


  % plotting the figures with figures function
  figures(theta_deg,theta_deg_nlg,altitude,color,phi_deg_nlg,psi_deg,gamma,dec,…
  travelminute,V,qs,qw,s);
```

```
% landing lat-long correction due to the calculation change in
  % retrograte orbit depending on the side of the re-entry

  if pro=='QUICKEST'
    if time<=43200
       theta_deg(end,end)=(rad2deg(theta_e)-(theta_deg(end,end)-
rad2deg(theta_e)))+360;
    end;
    if time>43200
      if theta_deg(end,end)>360
        theta_deg(end,end)=theta_deg(end,end)-360;
      end;
    end;
  else
     if theta_deg(end,end)>360
        theta_deg(end,end)=theta_deg(end,end)-360;
     end;
  end;

  % landing lat-long assignment for the tilt change
  phi_deg_end= phi_deg(end,end);
  theta_deg_end= theta_deg(end,end);

  % rotating landing and entry coodinates in order to express with
  % non-tilted coordinates by rechange_entry and rechange_landing
  % functions
  [theta_deg_nll,phi_deg_nll] = rechange_landing(theta_deg_end, phi_deg_end,…
   x_deg, y_deg, z_deg);
  [theta_ne,phi_ne] = rechange_entry(theta_e, phi_e, x_deg, y_deg, z_deg);

  % saving the outputs to be displayed on the gui
  entrytimehr=totalhour; entrytimemin=totalminute;
  save('entrytimehr'); save('entrytimemin');

  fparad=max(gamma_e); fpadeg=rad2deg(max(gamma_e));
  save('fparad'); save('fpadeg');

  entrylat= rad2deg(phi_ne); entrylong=rad2deg(theta_ne);
  save('entrylat'); save('entrylong');

  skippedrad=s_end; skippeddeg=rad2deg(s_end);
  save('skippedrad'); save('skippeddeg');

  uncorrlat= phi_deg(end,end); uncorrlong=theta_deg(end,end);
  save('uncorrlat');    save('uncorrlong');
```

```
landlat=phi_deg_nll; landlong=theta_deg_nll;
save('landlat'); save('landlong');

finspeed= V(end,end); alt=altitude(end,end);
save('finspeed'); save('alt');

timerr=abs((timenew/60)-totalminute); please='READY...';
save('timerr');  save('please');

% analyzing possible entry errors
erroranalysis(phi_e,theta_e,fpadeg,Ve,s_end,pro,time,theta_deg_nlg,phi_deg_nlg,…
 x_deg, y_deg, z_deg,phi_ne,theta_ne);

%opening gui screen to display outputs
inputdata;
delete('*.mat'); % to get rid of extra files
```

function inputdata.m

```
function varargout = inputdata(varargin)

% INPUTDATA M-file for inputdata.fig
%      INPUTDATA, by itself, creates a new INPUTDATA or raises the existing
%      singleton*.
%
%      H = INPUTDATA returns the handle to a new INPUTDATA or the handle to
%      the existing singleton*.
%
%      INPUTDATA('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in INPUTDATA.M with the given input arguments.
%
%      INPUTDATA('Property','Value',...) creates a new INPUTDATA or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before inputdata_OpeningFunction gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to inputdata_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Copyright 2002-2003 The MathWorks, Inc.
```

```
% Edit the above text to modify the response to help inputdata

% Last Modified by GUIDE v2.5 16-Jan-2008 16:40:01

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @inputdata_OpeningFcn, ...
                   'gui_OutputFcn',  @inputdata_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before inputdata is made visible.
function inputdata_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to inputdata (see VARARGIN)

% Choose default command line output for inputdata
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes inputdata wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = inputdata_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
```

```
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

load('entrytimehr');
set(handles.oentrytimehr,'string',num2str(entrytimehr));

load('entrytimemin');
set(handles.oentrytimemin,'string',num2str(entrytimemin));

load('fparad');
set(handles.ofparad,'string',num2str(fparad));

load('fpadeg');
set(handles.ofpadeg,'string',num2str(fpadeg));

load('entrylat');
set(handles.oentrylat,'string',num2str(entrylat));

load('entrylong');
set(handles.oentrylong,'string',num2str(entrylong));

load('skippedrad');
set(handles.oskippedrad,'string',num2str(skippedrad));

load('skippeddeg');
set(handles.oskippeddeg,'string',num2str(skippeddeg));

load('uncorrlat');
set(handles.ouncorrlat,'string',num2str(uncorrlat));

load('uncorrlong');
set(handles.uncorrlong,'string',num2str(uncorrlong));

load('landlat');
set(handles.olandlat,'string',num2str(landlat));

load('landlong');
set(handles.olandlong,'string',num2str(landlong));

load('finspeed');
set(handles.ofinspeed,'string',num2str(finspeed));
```

```
load('alt');
set(handles.oalt,'string',num2str(alt));

load('timerr');
set(handles.otimerr,'string',num2str(timerr));

load('please');
set(handles.please,'string',please);


function elat_Callback(hObject, eventdata, handles)
% hObject    handle to elat (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of elat as text
%        str2double(get(hObject,'String')) returns contents of elat as a double
lat=str2num(get(hObject,'String'));
save('lat');

% --- Executes during object creation, after setting all properties.
function elat_CreateFcn(hObject, eventdata, handles)
% hObject    handle to elat (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end



function elong_Callback(hObject, eventdata, handles)
% hObject    handle to elong (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of elong as text
%        str2double(get(hObject,'String')) returns contents of elong as a double
long=str2num(get(hObject,'String'));
```

```
save('long');

% --- Executes during object creation, after setting all properties.
function elong_CreateFcn(hObject, eventdata, handles)
% hObject    handle to elong (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end



function ehour_Callback(hObject, eventdata, handles)
% hObject    handle to ehour (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ehour as text
%        str2double(get(hObject,'String')) returns contents of ehour as a double
ho=str2num(get(hObject,'String'));
save('ho');

% --- Executes during object creation, after setting all properties.
function ehour_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ehour (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end



function emin_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to emin (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of emin as text
%        str2double(get(hObject,'String')) returns contents of emin as a double
min=str2num(get(hObject,'String'));
save('min');

% --- Executes during object creation, after setting all properties.
function emin_CreateFcn(hObject, eventdata, handles)
% hObject    handle to emin (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end



function esec_Callback(hObject, eventdata, handles)
% hObject    handle to esec (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of esec as text
%        str2double(get(hObject,'String')) returns contents of esec as a double
sec=str2num(get(hObject,'String'));
save('sec');

% --- Executes during object creation, after setting all properties.
function esec_CreateFcn(hObject, eventdata, handles)
% hObject    handle to esec (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
```

```
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end



function ex_Callback(hObject, eventdata, handles)
% hObject    handle to ex (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ex as text
%        str2double(get(hObject,'String')) returns contents of ex as a double
x=str2num(get(hObject,'String'));
save('x');

% --- Executes during object creation, after setting all properties.
function ex_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ex (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end



function ey_Callback(hObject, eventdata, handles)
% hObject    handle to ey (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ey as text
%        str2double(get(hObject,'String')) returns contents of ey as a double
y=str2num(get(hObject,'String'));
save('y');

% --- Executes during object creation, after setting all properties.
function ey_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ey (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end



function ez_Callback(hObject, eventdata, handles)
% hObject    handle to ez (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ez as text
%        str2double(get(hObject,'String')) returns contents of ez as a double
z=str2num(get(hObject,'String'));
save('z');

% --- Executes during object creation, after setting all properties.
function ez_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ez (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end


% --- Executes on button press in run.
function run_Callback(hObject, eventdata, handles)
% hObject    handle to run (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
wait='please wait...';
set(handles.please,'string',wait);
```

```
uiresume(inputdata);

% --- Executes on button press in restart.
function restart_Callback(hObject, eventdata, handles)
% hObject    handle to restart (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
reentry;


% --- Executes on selection change in popupmenu1.
function popupmenu1_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns popupmenu1 contents as cell array
%        contents{get(hObject,'Value')} returns selected item from popupmenu1

pro=get(hObject,'Value');
save('pro');

% --- Executes during object creation, after setting all properties.
function popupmenu1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
```

function entryoption.m

```
function [theta_e,phi_e,s_end] = entryoption(rot,center_theta,phi_l,theta_l,inc,pro)

if pro=='QUICKEST';
    phi_e=-inc;                          % inclination equals to entry latitude
```

```
% This part is determining the entry side depending on the landing place

    if rot<pi/2;
       theta_e=center_theta-pi/2;
    elseif rot>=pi/2 & rot<pi
       theta_e=center_theta+pi/2;
    elseif rot>=pi & rot<3*pi/2
       theta_e=center_theta-pi/2;
    else
       theta_e=center_theta+pi/2;
    end;
else
    if rot<pi/2;
       theta_e=center_theta+pi/2;
       phi_e=inc;                          % inclination equals to entry latitude
    elseif rot>=pi/2 & rot<pi
       theta_e=center_theta-pi/2;
       phi_e=inc;                          % inclination equals to entry latitude
    elseif rot>=pi & rot<3*pi/2
       theta_e=center_theta-pi/2;
       phi_e=-inc;                          % inclination equals to entry latitude
    else
       theta_e=center_theta+pi/2;
       phi_e=-inc;                          % inclination equals to entry latitude
    end;
end

% Finding total angular flight distance
s_end=acos(sin(phi_l)*sin(phi_e)+cos(phi_l)*cos(phi_e)*cos(abs(theta_e-theta_l)));

% angular distance adjustment
if pro=='QUICKEST'
  if rot>pi/2 & rot<3*pi/2;
     s_end=s_end;
  else
     s_end=(2*pi)-s_end;
  end;
else
  if rot<pi/2;
     s_end=(2*pi)-s_end;
  elseif rot>=pi/2 & rot<pi;
     s_end=(2*pi)+s_end;
  elseif rot>=pi & rot<3*pi/2;
     s_end=s_end;
  elseif  rot>=3*pi/2;
```

```
      s_end=(2*pi)-s_end;
   end;
end
```

function tiltchange.m

```
function [theta_l,phi_l] = tiltchange(theta_l_deg, phi_l_deg, x_deg, y_deg, z_deg);

fi_l=deg2rad(phi_l_deg);            %landing latitude
lambda_l=deg2rad(theta_l_deg);      %landing longitude

h_l=0;               % altitude
a=6378.1;            % semi major earth axis (ellipsoid equatorial radius)
b=6378.1;            % semi minor earth axis (ellipsoid polar radius)
f=(a-b)/a;           % flattening
e=sqrt(2*f-f^2);     % eccentricity

% changing geodedic coordinates to ECEF coordinaates
N=a/sqrt(1-((e^2)*(sin(fi_l))^2));

X_l=(N+h_l)*cos(fi_l)*cos(lambda_l);
Y_l=(N+h_l)*cos(fi_l)*sin(lambda_l);
Z_l=(N*(1-e^2)+h_l)*sin(fi_l);

ECEF_l=[X_l Y_l Z_l];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
% finding rotation matrix
x_r=deg2rad(x_deg);
y_r=deg2rad(y_deg);
z_r=deg2rad(z_deg);

R1= [1      0          0       ;
        0    cos(x_r)   -sin(x_r)  ;
        0    sin(x_r)    cos(x_r)] ;



R2= [cos(y_r)    0    sin(y_r)  ;
             0        1      0       ;
      -sin(y_r)    0    cos(y_r)] ;
```

```
R3= [cos(z_r)  -sin(z_r)    0   ;
        sin(z_r)   cos(z_r)    0   ;
                0        0        1 ] ;

R123=R1*R2*R3;          % rotation matrix

ECEF_nl= R123*ECEF_l';   % rotating the ECEF coordinates
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% changing ECEF coordinates to geodedic coordinaates
X_nl=ECEF_nl(1);
Y_nl=ECEF_nl(2);
Z_nl=ECEF_nl(3);

p=sqrt((X_nl^2)+(Y_nl^2));
teta=atan((Z_nl*a)/(p*b));
e1=sqrt((a^2-b^2)/b^2);

fi_nl=atan((Z_nl+e1^2*b*(sin(teta)^3))/(p-e^2*a*(cos(teta)^3)));
lambda_nl=atan2(Y_nl,X_nl);
h_nl=(p/cos(teta))-N;

if lambda_nl<0
   lambda_nl= lambda_nl+2*pi;
end

fi_nl_deg=rad2deg(fi_nl);
lambda_nl_deg=rad2deg(lambda_nl);

GEO=[fi_nl_deg lambda_nl_deg h_nl];

theta_l=lambda_nl;
phi_l=fi_nl;
```

function somver.m

```
function dx = solver(s,x,Br,sigma,lift_to_drag,mu,m,Beta,rho_s,S,Cd);

%this function is used to solve exact solution for reentry

 eta=x(1)/sqrt(Br);
 altitude=1/-Beta*log(eta*2*m*Beta/rho_s/S/Cd);
 r=altitude+6378;

A = [ -Br*tan(x(5))      zeros(1,6);...
```

```
        -2*x(2)*sqrt(Br)/cos(x(5))*(1+lift_to_drag*cos(sigma)*tan(x(5))) zeros(1,6);...
         zeros(1,7);...
         zeros(1,7);...
         sqrt(Br)/cos(x(5))*lift_to_drag*cos(sigma) zeros(1,6);...
         sqrt(Br)/((cos(x(5)))^2)*lift_to_drag*sin(sigma) zeros(1,6);...
         zeros(1,7)];


B= [0 ;...
       -x(2)*sin(x(5))/cos(x(5));...
       cos(x(6))/cos(x(4));...
       sin(x(6));...
       1-(cos(x(5)))^2/x(2);...
       -cos(x(6))*tan(x(4));
       r*sqrt(r)/sqrt(x(2)*mu)];

dx =A*x+B;

return
```

function changegraph.m

```
function [theta_deg_nlg,phi_deg_nlg] = change_graph(phi_l,theta, phi, theta_e,…
 x_deg, y_deg, z_deg, time,pro);

% this function is used to change the index numbers on the graph

fi_lg=phi;              %landing latitude
lambda_lg=theta;        %landing longitude

if pro=='QUICKEST'
  if time<=43200
        lambda_lg=(theta_e-(lambda_lg-theta_e))+2*pi;
  end;
  if time>43200
     for j=1:length(phi);
       if lambda_lg(j,1)>2*pi;
        lambda_lg(j,1)=lambda_lg(j,1)-2*pi;
       end;
     end;
  end;
else
     for j=1:length(phi);
       if lambda_lg(j,1)>2*pi;
        lambda_lg(j,1)=lambda_lg(j,1)-2*pi;
```

```
        end;
      end;

end;

% constants
h_l=0;          % altitude
a=6378.1;        % semi major earth axis (ellipsoid equatorial radius)
b=6378.1;        % semi minor earth axis (ellipsoid polar radius)
f=(a-b)/a;       % flattening
e=sqrt(2*f-f^2);  % eccentricity

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% finding rotation matrix
x_rg=deg2rad(-x_deg);
y_rg=deg2rad(-y_deg);
z_rg=deg2rad(-z_deg);

R1g= [1     0          0        ;
       0   cos(x_rg)   -sin(x_rg)  ;
       0   sin(x_rg)    cos(x_rg)] ;

R2g= [cos(y_rg)    0    sin(y_rg)  ;
           0         1     0        ;
       -sin(y_rg)    0    cos(y_rg)] ;

R3g= [cos(z_rg)  -sin(z_rg)    0   ;
        sin(z_rg)   cos(z_rg)    0   ;
             0        0          1 ] ;

R123g=R3g*R2g*R1g; % rotation matrix
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% changing geodedic coordinates to ECEF coordinaates
for i=1:length(phi);

N_lg(i,1)=a/sqrt(1-((e^2)*(sin(fi_lg(i)))^2));

X_lg(i,1)=(N_lg(i,1)+h_l)*cos(fi_lg(i))*cos(lambda_lg(i));
Y_lg(i,1)=(N_lg(i,1)+h_l)*cos(fi_lg(i))*sin(lambda_lg(i));
Z_lg(i,1)=(N_lg(i,1)*(1-e^2)+h_l)*sin(fi_lg(i));

ECEF_lg(i,:)=[X_lg(i,1) Y_lg(i,1) Z_lg(i,1)];
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% rotating the ECEF coordinates
ECEF_nlg(:,i)= R123g*ECEF_lg(i,:)';
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% changing ECEF coordinates to geodedic coordinaates
X_nlg(i,1)=ECEF_nlg(1,i);
Y_nlg(i,1)=ECEF_nlg(2,i);
Z_nlg(i,1)=ECEF_nlg(3,i);

p=sqrt((X_nlg(i,1)^2)+(Y_nlg(i,1)^2));
teta=atan((Z_nlg(i,1)*a)/(p*b));
e1=sqrt((a^2-b^2)/b^2);

fi_nlg(i,1)=atan((Z_nlg(i,1)+e1^2*b*(sin(teta)^3))/(p-e^2*a*(cos(teta)^3)));
lambda_nlg(i,1)=atan2(Y_nlg(i,1),X_nlg(i,1));
h_nlg(i,1)=(p/cos(teta))-N_lg(i,1);

end;

fi_nlg_deg=rad2deg(fi_nlg);
lambda_nlg_deg=rad2deg(lambda_nlg);

theta_deg_nlg=lambda_nlg_deg;
phi_deg_nlg=fi_nlg_deg;
```

function figures.m

```
function
figures(theta_deg,theta_deg_nlg,altitude,color,phi_deg_nlg,psi_deg,gamma,dec,…
travelminute,V,qs,qw,s)

 % figures
  figure(1);
  subplot(2,1,1);
  plot(theta_deg,122,'r-');
  hold on;
  plot(theta_deg,altitude,color(1));
  grid on;
  xlabel('Skipped Longitude(degrees)');
  ylabel('Altitude (km)');

  subplot(2,1,2);
  plot(s*6378,122,'r-');
```

```
hold on;
plot(s*6378,altitude,color(2));
grid on;
xlabel('Skipped Distance(km)');
ylabel('Altitude (km)');


figure(2);
subplot(2,1,1);
plot(travelminute,V, color(2));
grid on;
xlabel('Time (min)');
ylabel('Velocity (km/s)');

subplot(2,1,2);
plot(travelminute,altitude,color(1));
hold on;
plot(travelminute,122,'r-');
grid on;
xlabel('Time(min)');
ylabel('Altitude (km)');


figure(3);
plot(linspace(min(rad2deg(gamma)),max(rad2deg(gamma)),1000),122,'r-');
hold on;
plot(rad2deg(gamma),altitude,color(4));
grid on;
xlabel('Flight Path Angle-gamma (deg)');
ylabel('Altitude (km)');


figure(4);
plot3(theta_deg,phi_deg_nlg,altitude,color(3));
grid on;
title('3D reentry plot');
xlabel('Longitude(degrees)');
ylabel('Latitude(degrees)');
zlabel('Altitude (km)');


figure(5);
subplot(2,2,[1 3]);
plot(linspace(min(dec),max(dec),1000),122,'r-');
hold on;
```

```
plot(dec,altitude,color(1));
grid on;
xlabel('Deceleration (g)');
ylabel('Altitude (km)');

subplot(2,2,2);
plot(linspace(min(qs),max(qs),1000),122,'r-');
hold on;
plot(qs,altitude,color(3));
grid on;
xlabel('qs, stagnation heat flux');
ylabel('Altitude (km)');

subplot(2,2,4);
plot(linspace(min(qw),max(qw),1000),122,'r-');
hold on;
plot(qw,altitude,color(4));
grid on;
xlabel('qw, wall heat flux');
ylabel('Altitude (km)');


figure(6);
plot(travelminute,dec,'r');
grid on;
xlabel('Time(min)');
ylabel('Deceleration (g)');


figure(8);
load('topo.mat','topo','topomap1');
topo2 = [topo(:,181:360) topo(:,1:180)];
contour(-179:180,-89:90,topo2,[0 0],'b')
axis equal;
grid on;
set(gca,'XLim',[-180 180],'YLim',[-90 90], ...
'XTick',[ -180 :20: 180 ], ...
'Ytick',[ -90 :20: 90 ]);
hold on;
plot(theta_deg_nlg,phi_deg_nlg,'r','linewidth',2);
grid on;
hold on
plot(theta_deg_nlg(1,1),phi_deg_nlg(1,1),'go','linewidth',2);
hold on
plot(theta_deg_nlg(end,end),phi_deg_nlg(end,end),'k*','linewidth',2)
```

```
title('Reentry Ground Track');
xlabel('Longitude(degrees)');
ylabel('Latitude(degrees)');
```

function rechange_landing.m

```
function [theta_deg_nll,phi_deg_nll] = rechange_landing(theta_deg_end, …
phi_deg_end, x_deg, y_deg, z_deg);

fi_ll=deg2rad(phi_deg_end);                %landing latitude
lambda_ll=deg2rad(theta_deg_end);          %landing longitude

h_ll=0;              % altitude
a=6378.1;            % semi major earth axis (ellipsoid equatorial radius)
b=6378.1;            % semi minor earth axis (ellipsoid polar radius)
f=(a-b)/a;           % flattening
e=sqrt(2*f-f^2);  % eccentricity

% changing geodedic coordinates to ECEF coordinaates
N=a/sqrt(1-((e^2)*(sin(fi_ll))^2));

X_ll=(N+h_ll)*cos(fi_ll)*cos(lambda_ll);
Y_ll=(N+h_ll)*cos(fi_ll)*sin(lambda_ll);
Z_ll=(N*(1-e^2)+h_ll)*sin(fi_ll);

ECEF_ll=[X_ll Y_ll Z_ll];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% finding rotation matrix
x_r=deg2rad(-x_deg);
y_r=deg2rad(-y_deg);
z_r=deg2rad(-z_deg);

R1= [1      0          0       ;
       0    cos(x_r)   -sin(x_r)  ;
       0    sin(x_r)    cos(x_r)] ;

R2= [cos(y_r)    0    sin(y_r)  ;
           0        1      0       ;
      -sin(y_r)    0    cos(y_r)] ;

R3= [cos(z_r)  -sin(z_r)    0   ;
        sin(z_r)   cos(z_r)    0   ;
                0        0        1 ] ;
```

```
R123=R3*R2*R1;          % rotation matrix

ECEF_nll= R123*ECEF_ll';  % rotating the ECEF coordinates
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% changing ECEF coordinates to geodedic coordinaates
X_nll=ECEF_nll(1);
Y_nll=ECEF_nll(2);
Z_nll=ECEF_nll(3);

p=sqrt((X_nll^2)+(Y_nll^2));
teta=atan((Z_nll*a)/(p*b));
e1=sqrt((a^2-b^2)/b^2);

fi_nll=atan((Z_nll+e1^2*b*(sin(teta)^3))/(p-e^2*a*(cos(teta)^3)));
lambda_nll=atan2(Y_nll,X_nll);
h_nll=(p/cos(teta))-N;

if lambda_nll<0
   lambda_nll= lambda_nll+2*pi;
end

fi_nll_deg=rad2deg(fi_nll);
lambda_nll_deg=rad2deg(lambda_nll);

GEO=[fi_nll_deg lambda_nll_deg h_nll];

theta_deg_nll=lambda_nll_deg;
phi_deg_nll=fi_nll_deg;
```

function rechange_entry.m

```
function [theta_ne,phi_ne] = rechange_entry(theta_e, phi_e, x_deg, y_deg, z_deg);

% this function is used to rotate the coordinates according to the tilt
% angle of the earth
fi_e=phi_e;             % entry latitude
lambda_e=theta_e;       % entry longitude

%constants
h_l=0;              % altitude
a=6378.1;           % semi major earth axis (ellipsoid equatorial rdius)
b=6378.1;           % semi minor earth axis (ellipsoid polar rdius)
```

```
f=(a-b)/a;           % flattening
e=sqrt(2*f-f^2);  % eccentricity

% changing geodedic coordinates to ECEF coordinaates
N=a/sqrt(1-((e^2)*(sin(fi_e))^2));

X_e=(N+h_l)*cos(fi_e)*cos(lambda_e);
Y_e=(N+h_l)*cos(fi_e)*sin(lambda_e);
Z_e=(N*(1-e^2)+h_l)*sin(fi_e);

ECEF_e=[X_e Y_e Z_e];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% finding the rotation matrix
x_r=deg2rad(-x_deg);
y_r=deg2rad(-y_deg);
z_r=deg2rad(-z_deg);

R1= [1      0          0      ;
       0   cos(x_r)   -sin(x_r)  ;
       0   sin(x_r)    cos(x_r)] ;

R2= [cos(y_r)    0    sin(y_r)  ;
           0        1      0       ;
      -sin(y_r)    0    cos(y_r)] ;

R3= [cos(z_r)  -sin(z_r)    0   ;
        sin(z_r)   cos(z_r)    0   ;
              0         0        1 ] ;

R123=R3*R2*R1;         % rotation matrix

ECEF_ne= R123*ECEF_e'; % rotating ECEF coordinates
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% changing ECEF coordinates to geodedic coordinaates
X_ne=ECEF_ne(1);
Y_ne=ECEF_ne(2);
Z_ne=ECEF_ne(3);

p=sqrt((X_ne^2)+(Y_ne^2));
teta=atan((Z_ne*a)/(p*b));
e1=sqrt((a^2-b^2)/b^2);

fi_ne=atan((Z_ne+e1^2*b*(sin(teta)^3))/(p-e^2*a*(cos(teta)^3)));
```

```
lambda_ne=atan2(Y_ne,X_ne);
h_ne=(p/cos(teta))-N;

if lambda_ne<0
   lambda_ne= lambda_ne+2*pi;
end

fi_ne_deg=rad2deg(fi_ne);
lambda_ne_deg=rad2deg(lambda_ne);

GEO=[fi_ne_deg lambda_ne_deg h_ne];

theta_ne=lambda_ne;
phi_ne=fi_ne;
```

Bibliography

1. Z. R. Putnam, R. D. Braun, S. H. Bairstow and G. H. Barton "Improving Lunar Return entry Footprints Using Enhanced skip Trajectory Guidance," AIAA 2006-7438

2. Z. R. Putnam, R. D. Braun, R. R. Rohrschneider and J. A. Dec "Entry System Options for Human Return from the Moon and Mars," AIAA 2005-5915

3. Hicks, Kerry D. *Introduction to Atmospheric Reentry*, Unpublished Text Book, Air Force Institute of Technology, WPAFB, 2007

4. Harpold, J. C., and Graves, C. A., "Shuttle Entry Guidance," Journal of the Astronautical Sciences, Vol. 27, No. 3, Jul-Sept. 1979, pp. 139-268

5. R.R. Costa, J.A. Silva, S.F. Wu, Q.P. Chu and J.A Mulder "Atmospheric Entry Modeling and Simulation," Journal of the Astronautical Sciences, Vol. 39, No. 4 AIAA-3855-668

6. Ashok Joshi, K. Sivan and S. Savithri Amma "Predictor-Corrector Reentry Guidance Algorithm with Path Constraints for Atmospheric Entry Vehicles," AIAA-26306-656

7. S. H. Bairstow and G. H. Barton " Orion Reentry Guidance with Extended Range Capability Using PredGuid," AIAA-2007-6427

8. R.T. Bilbeau and D.S. Rubanstein “Trajectory Optimization for a Fixed-Trim Reentry Vehicle Using Direct Collocation and Nonlinear Programming “, AIAA-200-4262-434

9. J. R. Rea and Z. R. Putnam “ Comparison of Two Orion Skip Entry Guidance Algorithms”, AIAA-2077-6424

10. *NASA Exploration Systems Architecture Study (ESAS),* November 2005

11. Vinh, N. X., and Brace, F. C. “Qualitative and Quantitative Analysis of the Exact Atmospheric Entry Equations Using Chapman’s Variables,” IAF Paper No. 74-010, Presented at the XXVth Congress of the International Astronautical Federation, Amsterdam, The Netherlands, Oct. 1974.

12. Loh, W. H. T., *Dynamics and Thermodynamics of Planetary Entry*, Englewood Cliffs, NJ: Prentice-Hall, Inc., 1963.

13. W. E. Wiesel, *Space Flight Dynamics*, The McGraw Hill Companies, Inc., 1997

14. W. J. Larson and J. R. Wertz, *Space Mission Analysis and Design*, Microcosm, Inc. and W. J. Larson , 2005

15. World Wide Web

http://www1.elsevier.com/homepage/saa/eccc3/paper48/img88.gif

16. M. Laftery and T. Fox, “The Crew exploration Vehicle (CEV) and The Next Generation of Human Spaceflight,” Space Exploration Systems, Boeing Company, USA, 10 April 2007

17. B. Bowring, “Transformation from Spatial to Geographical Coordinates,” Survey Review, XXIII, 1976

18. World Wide Web
http://www.aerospaceweb.org/question/astronomy/moon/orbit.jpg

19. Photo courtesy Google Earth™ mapping service

20. World Wide Web, “NASA Fact Sheets, Moon Fact Sheet”
http://nssdc.gsfc.nasa.gov/planetary/factsheet/moonfact.html

21. World Wide Web
http://upload.wikimedia.org/wikipedia/commons/thumb/9/90/Skip_reentry_trajectory.svg/800px-Skip_reentry_trajectory.svg.png

Vita

First Lieutenant Emre Kaya graduated from Kuleli Military high School in Istanbul, Turkey. He entered Turkish Air Force Academy in Istanbul, where he graduated with a Bachelor of Science degree in Electronics Engineering in August 2002.

His first assignment was in Izmir, Turkey as a trainee at pilot training school in 2002. In July 2005, he was assigned to the 152nd squadron, 5th Main Jet Base, Merzifon where he served as an F-16 pilot. In September 2006, he entered Graduate School of Space Systems, Air Force Institute of Technology.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From – To)*
March 2008	Master's Thesis	Sep 2006 – Mar 2008

4. TITLE AND SUBTITLE

CREW EXPLORATION VEHICLE (CEV) SKIP ENTRY TRAJECTORY

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Kaya, Emre., First Lieutenant, TuAF

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S)

Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way, Building 640
WPAFB OH 45433-8865

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT/GSS/ENY/08-M06

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This research effort develops a program using MATLAB® to solve the equations of motion for the atmospheric reentry of the Crew Exploration Vehicle (CEV) which is assumed to be in the phase of a lunar return trajectory that could be initiated any time during the mission. The essential reason for this research is to find a solution for the problem of an unplanned lunar return in addition to the normal procedures. Unlike Apollo type missions, the CEV would still be able to land on any preplanned available landing sites without any additional delay. In Apollo type missions, the return phase had to be initiated in a restricted time window so that the crew module could enter the atmosphere at the preplanned time and be able to land at the planned landing site. Using skip entry procedures, landing location and time will be more accurate in addition to having the time flexibility for reentry. This MATLAB® program is designed to find the reentry parameters for given landing location according to the current alignment of the moon using a lunar return speed including the atmospheric trajectory of the CEV.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Kerry Hicks, Lt Col, USAF
U	U	U	UU	118	19b. TELEPHONE NUMBER *(Include area code)* (937) 255-3636, ext 4568, Email: kerry.hicks@afit.edu